SDK_API

The “WinQHYCAM.h” is the function prototype declaration file. The “WinQHYCAM.dll” is the dynamic link library. And the static link library is “WinQHYCAM.lib”.

Define Declaration

The parameters of camera type：

#define DEVICETYPE_IMG0S 00

#define DEVICETYPE_IMG0H 01

#define DEVICETYPE_IMG0L 02

#define DEVICETYPE_IMG0X 03

#define DEVICETYPE_IMG1S 10

#define DEVICETYPE_IMG2S 20

#define DEVICETYPE_IMG1E 110

#define DEVICETYPE_IMG2P 220

#define DEVICETYPE_QHY8L 385

#define DEVICETYPE_QHY9S 390

#define DEVICETYPE_QHY16 16

#define DEVICETYPE_QHY5 51

#define DEVICETYPE_QHY5II 52

#define DEVICETYPE_QHY5LII 53

#define DEVICETYPE_QHY5TII 54

#define DEVICETYPE_QHY5RII 55

#define DEVICETYPE_QHY5PII 56

#define DEVICETYPE_QHY5VII 57

#define DEVICETYPE_Q16000 160

#define DEVICETYPE_QHY22 694

#define DEVICETYPE_QHY6 60

#define DEVICETYPE_UNKNOWN -1

The function CONTROL：

enum CONTROL_ID

{

CONTROL_BRIGHTNESS=0, //control the brightness (range: 0-100 ,default: 0) .

CONTROL_CONTRAST=1, //control the contrast (range: 0.0-2.0 , default: 1.0).

CONTROL_WBR=2, //control the red value of white balance (range: 0-63(default) , 50-200(QHY5LII) , 64-255(IMG132E)).

CONTROL_WBB=3, //control the blue value of white balance (range: 0-63(default) , 50-200(QHY5LII) , 64-255(IMG132E)).

```
    CONTROL_WBG=4,        //control the green value of white balance (range: 0-63(default)  , 50-200(QHY5LII) , 64-255(IMG132E)).
    CONTROL_GAMMA=5,      //control the gamma (range: 0.0-2.0, default: 1.0).
    CONTROL_GAIN=6,       //control the gain (range: 0-1000).
    CONTROL_OFFSET=7,     //control the offset (range: 0-255).
    CONTROL_EXPOSURE=8,       //control the exposure time.
    CONTROL_SPEED=9,          //control the camera transfer speed.  if the value is '0' means low speed transmission, if  '1' means high speed.
    CONTROL_TRANSFERBIT=10,       //control the transfer bit ( 8 bit or 16 bit).
    CONTROL_USBTRAFFIC=11,    //control the USB bandwidth,(only for QHY5II and QHY5LII to reduce the USB bandwidth . Value range: (30-0), 0 - USB transmission is the fastest , 30 - transmission is the slowest)
    CONTROL_ROWNOISERE=12,        //control the row noise (only for QHY5II to eliminate the row noise)
    CONTROL_HDRMODE=13        //control the HDR mode (only for QHY5LII to control the HDR mode open or close)
};

Bayer pattern :
enum BAYER_ID
{
    BAYER_GB=1,
```

```
    BAYER_GR=2,
    BAYER_BG=3,
    BAYER_RG=4
};
```

Type of smoothing：

```
enum SMOOTHTYPE
{
    BLUR_NO_SCALE=0,
    BLUR=1,
    GAUSSIAN=2,
    MEDIAN=3,
    BILATERAL=4
};
```

Function Declaration

1. **Name**： OpenCameraById

Prototype： extern "C" int __stdcall OpenCameraById(int id);

Function： Connect to the specified camera and init it.

Parameters： id - The camera type.

Returned : If success call, return the type of camera connected to PC. Otherwise, return DEVITYPE_UNKNOWN.

Instruction : This function can open the specified camera. It apply to connect one single camera fast. And this API function is the premise of all camera control function.

2. **Name** : CloseCamera

Prototype : extern "C" void __stdcall CloseCamera(int id);

Function : Close the specified camera.

Parameters : id - The type of camera.

Returned :

Instruction :

3. **Name** : BeginLive

Prototype : extern "C" void __stdcall BeginLive();

Function : Start to exposure.

Parameters :

Returned :

Instruction : To some types of cameras, it just do once, then will start to continuous mode. But to the cameras (IMG2P,QHY8L,QHY9S,QHY22,QHY16,QHY6) you need call this

function firstly when you do the exposure to get the image every time.

4. **Name** : GetROIImageData

Prototype : extern "C" void __stdcall GetROIImageData(int ow, int oh, int width, int height, int bpp, int channels, unsigned char *rawArray);

Function : Get the tailored image

Parameters :

ow - The upper left corner X coordinates of the ROI image.

oh - The upper left corner X coordinates of the ROI image.

width - The width of ROI image.

height - The height of ROI image.

bpp - The bit depth of ROI image.

channels - The channels number of ROI image.

rawArray - The memory space to store the ROI image.

Returned :

Instruction : This function is used to get the image data to memory space and has the cutting function. It support all types of cameras. Notice: ow+width <= w (the width of resolution) ; oh + height <= h (the height of resolution).

5. **Name** : GetImageData

Prototype : extern "C" void __stdcall GetImageData(int w, int h, int bpp, int channels, unsigned char *rawArray);

Function : Get the image.

Parameters : width - The width of image.

height - The height of image.

bpp - The bit depth of image.

Channels - The channels number of image.

rawArray - The memory space to store the image data.

Returned :

Instruction : This function is used to get the image data to memory space and has the cutting function. It support all types of cameras.

6. **Name** : StopLive

Prototype : extern "C" void __stdcall StopLive();

Function : Stop the exposure.

Parameters :

Returned :

Instruction : This function is the camera to stop exposure. It support all types of cameras.

7. **Name** : SetBin

Prototype : extern "C" void __stdcall SetBin(int w, int h);

Function : Set merger mode.

Parameters : w - width.

h - height.

Returned :

Instruction : It will merge to pixel with image data collected.

8. **Name** : IsColorCam

Prototype : extern "C" bool __stdcall IsColorCam();

Function : Query whether the camera is a color camera.

Parameters :

Returned : If it is color camera,return true. Otherwise,return false.

Instruction :

9. **Name** : IsAstroCcd

Prototype : extern "C" bool __stdcall IsAstroCcd(int m_devType);

Function : Query whether it is astronomical CCD camera.

Parameters : m_devType - The type of camera.

Returned : If it is astronomical CCD, return true. If not,return false.

Instruction : Whether it is astronomical CCD camera (means whether it support refrigeration). If it is astronomical CCD,it support refrigeration. Otherwise, not support.

10. **Name** : GetMaxFrameLength

Prototype : extern "C" int __stdcall GetMaxFrameLength();

Function : Get the image minimum required memory space.

Parameters :

Returned : Size of memory space(Bytes).

Instruction :

11. **Name** : SetResolution

Prototype : extern "C" void __stdcall SetResolution(int x, int y);

Function : Set the resolution of image.

Parameters : x - width.

y - height.

Returned :

Instruction : Set the resolution of the camera collection.

12. **Name** : GetImageFormat

Prototype : extern "C" void __stdcall GetImageFormat(int *w, int *h, int *bpp, int *channels);

Function : Get the maximum resolution,image depth,channl number that camera support.

Parameters : w - The width of image.

h - The height of image.

bpp - The bit depth of image.

channels - The channel number of image.

Returned :

Instruction :

13. **Name** : Bit16To8_Stretch

Prototype : extern "C" void __stdcall Bit16To8_Stretch(unsigned char *InputData16, unsigned char *OutputData8, int imageX, int imageY, unsigned short B, unsigned short W);

Function : The 16-bit data stretch into 8 bits data.

Parameters : InputData16 - The inputted 16-bit image data.

OutputData8 - The outputted 8-bit image data.

imageX - The width of image.

imageY - The height of image.

B - Gray stretch Black.

W - Gray stretch White.

Returned :

Instruction : We suggest that you would show the histogram of 16-bit before call the function. Then you can confirm the value of B and W from the histogram.

14. **Name** : IsHighSpeed

Prototype : extern "C" bool __stdcall IsHighSpeed();

Function : Query whether the camera support high speed mode.

Parameters :

Returned : If it support,return true. otherwise,return false.

Instruction :

15. **Name** : HistInfo

Prototype : extern "C" void __stdcall HistInfo(int x, int y, unsigned char *InBuf, unsigned char *outBuf, bool enDebug);

Function : Display the inputed 16 RAW data in the histogram (the output resolution is 192x130).

Parameters : x - The width of inputed image.

y - The height of inuptted image.

InBuf - The inuptted image data.

outBuf - The outputted image data.

enDebug - use for debug. enDebug=true, will show the histogram

Returned :

Instruction :

16. **Name** : Smooth

Prototype : extern "C" void __stdcall Smooth(int x, int y, int bpp, unsigned char *InBuf, unsigned char *outBuf, bool enDebug);

Function : Filter the inuptted 16 RAw data

Parameters : x - The width of inuptted image.

y - The height of inuptted image.

bpp - The bit depth of inuptted image.

InBuf - The inuptted image data.

outBuf - outputted image data.

enDebug - use for debug. enDebug=true, will show the output image.

Returned :

Instruction :

17. **Name** : IsControlAvailable

Prototype : extern "C" bool __stdcall IsControlAvailable(CONTROL_ID id);

Function： Query whether the CONTROL is available

Parameters：id - The id of CONTROL.

Returned： If it is available,return true. else return false.

Instruction： This API function will query whether the CONTROL is available for the camera. It support all cameras. And the parameter ‘id’ is the id number of CONTROL which in enum CONTROL_ID.

18. **Name**：GetMinValue

Prototype：extern "C" double __stdcall GetMinValue(CONTROL_ID id);

Function：Get the minimum value of the CONTROL.

Parameters：id - The id of CONTROL

Returned： The minimum value

Instruction： This function can get the minimum value of the CONTROL that you can set. The API function support all types of cameras.

19. **Name**：GetMaxValue

Prototype：extern "C" double __stdcall GetMaxValue(CONTROL_ID id);

Function：Get the maximum value of the CONTROL

Parameters：id - The id of Camera

Returned： The maximum value

Instruction ： This function can get the maximum value of the CONTROL that you can set. The API function support all types of cameras.

20. **Name** : GetPattern

Prototype ： extern "C" int __stdcall GetPattern();

Function ： Get the Bayer Pattern of camera

Parameters :

Returned ： The Bayer Pattern code

Instruction ： The Bayer Pattern code returned can find the Bayer Pattern in enum BAYER_ID.

21. **Name** : SetValue

Prototype ： extern "C" void __stdcall SetValue(CONTROL_ID id, double value);

Function ： Set the CONTROL value

Parameters ： id - The id of CONTROL

value - The value you wanted to set.

Returned :

Instruction ： This function support all CONTROL. You just need to pass the id of CONTROL and the value you want to set.

22. **Name** : GetValue

Prototype : extern "C" double __stdcall GetValue(CONTROL_ID id);

Function : Get the CONTROL value

Parameters : id - The id of CONTROL

Returned : The value of CONTROL

Instruction : This function can get the value of CONTROL that you setted. It support all CONTROL.

23. **Name** : GuiderControl

Prototype : extern "C" void __stdcall GuideControl(unsigned char Direction, unsigned short PulseTime);

Function : Control the star guide port.

Parameters : Direction - The controlled direction(range: 0-3).

PulseTime - Delay time(unit:ms).

Returned :

Instruction :

24. **Name** : ImageSmooth

Prototype : extern "C" void __stdcall ImageSmooth(int x, int y, int bpp, int channels, unsigned char *src, unsigned char *dst, int smoothtype=GAUSSIAN, int size1=3, int size2=0);

Function : Filter the image.

Parameters :

x - The width of image.

y - The height of image.

bpp - The bit depth of image.

channels - The channels of image.

src - The source image.

dst - The destination image.

smoothtype - Type of the smoothing:

BLUR_NO_SCALE : linear convolution with size1 X size2 box kernel (all 1’ s).

BLUR linear : convolution with size1 X size2 box kernel (all 1’ s) with subsequent scaling by 1/(size1 * size2).

BLUR linear : convolution with size1 X size2 box kernel (all 1’ s) with subsequent scaling by 1/(size1 * size2).

GAUSSIAN : linear convolution with a size1 X size2 Gaussian kernel.

MEDIAN : median filter with a size1 X size1 square aperture.

BILATERAL : bilateral filter with a size1 X size1 square aperture.

size1 - The first parameter of the smoothing operation, the aperture width. Must be a positive odd number (1, 3, 5, ...)

size2 - The second parameter of the smoothing operation, the aperture height. Ignored by MEDIAN and BILATERAL methods. In the case of simple scaled/non-scaled and Gaussian blur if size2 is zero, it is set to size1 . Otherwise it must be a positive odd number.

Returned :

Instruction :

25. **Name** : ImageSharpen

Prototype : extern "C" void __stdcall ImageSharpen(int x, int y, int bpp,int channels,unsigned char *src, unsigned char *dst, int sTemplate);

Function : Sharpen the image

Parameters :

x - The width of image.

y - The height of image.

bpp - The bit depth of image.

channels - The channels of image.

src - The source image.

dst - The destination image.

sTemplate - the template of sharpen:(value range: 1-4)

value '1' : (3x3 template)

0	-1	0
-1	5	-1
0	-1	0

value '2' : (3x3 template)

-1	-1	-1
-1	9	-1
-1	-1	-1

value '3' : (5x5 template)

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	17	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

value '4' : (5x5 template)

-1	-1	-1	-1	-1
-1	-2	-2	-2	-1
-1	-2	33	-2	-1
-1	-2	-2	-2	-1
-1	-1	-1	-1	-1

Returned :

Instruction :

26. **Name** : AutoTempControl

Prototype : extern "C" double __stdcall AutoTempControl(double TargetTemp);

Function : Control the camera temperature

Parameters : TargetTemp - The target temperature.

Returned : Current temperature

Instruction :