IMG_256

QHYCCD C WRAPPER API

说明文档

V1.0

目录

[1. 项目介绍 1](#_Toc29685)

[2. 函数介绍 2](#_Toc2404)

[2.1. C\_InitQHYCCDResource 2](#_Toc19878)

[2.2. C\_ReleaseQHYCCDResource 2](#_Toc13090)

[2.3. C\_ScanQHYCCD 2](#_Toc27672)

[2.4. C\_GetQHYCCDId 3](#_Toc17514)

[2.5. C\_OpenQHYCCD 3](#_Toc13189)

[2.6. C\_CloseQHYCCD 4](#_Toc14575)

[2.7. C\_GetQHYCCDReadModeNumber 4](#_Toc16305)

[2.8. C\_GetQHYCCDReadModeName 4](#_Toc30935)

[2.9. C\_SetQHYCCDReadMode 5](#_Toc28119)

[2.10. C\_SetQHYCCDStreamMode 5](#_Toc21364)

[2.11. C\_InitQHYCCD 6](#_Toc8946)

[2.12. C\_SetQHYCCDDebayerOnOff 6](#_Toc2276)

[2.13. C\_SetQHYCCDParam\_Bits 7](#_Toc6521)

[2.14. C\_SetQHYCCDBinMode 7](#_Toc8553)

[2.15. C\_SetQHYCCDResolution 8](#_Toc6593)

[2.16. C\_SetQHYCCDChipInfo 8](#_Toc2227)

[2.17. C\_GetQHYCCDEffectiveArea 9](#_Toc23199)

[2.18. C\_GetQHYCCDOverScanArea 10](#_Toc21635)

[2.19. C\_GetQHYCCDMemLength 11](#_Toc29707)

[2.20. C\_SetQHYCCDParam\_Exposure 11](#_Toc6472)

[2.21. C\_SetQHYCCDParam\_Gain 12](#_Toc23495)

[2.22. C\_SetQHYCCDParam\_Offset 12](#_Toc1179)

[2.23. C\_SetQHYCCDParam\_Traffic 12](#_Toc25691)

[2.24. C\_SetQHYCCDParam\_WBR 13](#_Toc29909)

[2.25. C\_SetQHYCCDParam\_WBG 13](#_Toc14571)

[2.26. C\_SetQHYCCDParam\_WBB 14](#_Toc15884)

[2.27. C\_SetQHYCCDParam\_Brightness 14](#_Toc25771)

[2.28. C\_SetQHYCCDParam\_Contrast 15](#_Toc4154)

[2.29. C\_SetQHYCCDParam\_Gamma 15](#_Toc11003)

[2.30. C\_ExpQHYCCDSingleFrame 16](#_Toc20317)

[2.31. C\_GetQHYCCDSingleFrame 16](#_Toc23930)

[2.32. C\_CancelQHYCCDExposingAndReadout 17](#_Toc30156)

[2.33. C\_BeginQHYCCDLive 17](#_Toc1930)

[2.34. C\_GetQHYCCDLiveFrame 18](#_Toc6688)

[2.35. C\_StopQHYCCDLive 19](#_Toc17182)

[2.36. C\_GetQHYCCDTrigerInterfaceNumber 19](#_Toc15701)

[2.37. C\_GetQHYCCDTrigerInterfaceName 20](#_Toc26824)

[2.38. C\_SetQHYCCDTrigerInterface 20](#_Toc27417)

[2.39. C\_SetQHYCCDTrigerFunction 21](#_Toc7516)

[2.40. C\_EnableQHYCCDTrigerOut 21](#_Toc17757)

[2.41. C\_EnableQHYCCDBurstMode 21](#_Toc4528)

[2.42. C\_SetQHYCCDBurstModeStartEnd 22](#_Toc7587)

[2.43. C\_SetQHYCCDBurstModePatchNumber 23](#_Toc11574)

[2.44. C\_SetQHYCCDBurstIDLE 23](#_Toc4520)

[2.45. C\_ReleaseQHYCCDBurstIDLE 23](#_Toc25218)

[2.46. C\_GetQHYCCDParam\_CurTemperature 24](#_Toc3664)

[2.47. C\_SetQHYCCDParam\_TargetTemperature 24](#_Toc9476)

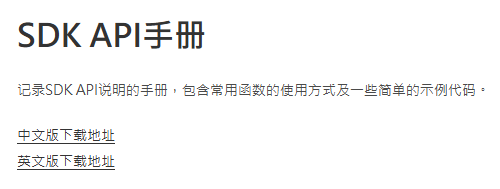
[2.48. C\_SetQHYCCDParam\_CoolerPWM 25](#_Toc29440)

# 

# 项目介绍

此工程围绕QHYCCD SDK（即qhyccd.dll库文件）进行开发，由于QHYCCD SDK使用C++语言开发，无法满足C语言直接调用QHYCCD SDK的需求，因此因此开发此工程，旨在提供简单实用的函数接口，以实现C程序调用QHYCCD SDK控制QHYCCD相机设备的需求。

关于QHYCCD SDK函数具体细节可以参考SDK API文档，下载位置：https://www.qhyccd.cn/sdk\_demo/。



# 函数介绍

注：QHYCCD\_API定义为\_\_declspec(dllexport)。

## C\_InitQHYCCDResource

**函数说明：**

初始化QHYCCD SDK内部资源，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为InitQHYCCDResource。

**函数原型：**

QHYCCD\_API uint32\_t C\_InitQHYCCDResource();

**示例代码：**

int ret = C\_InitQHYCCDResource();

if(ret == 0)

{

printf(“C\_InitQHYCCDResource() success\n”);

}

## C\_ReleaseQHYCCDResource

**函数说明：**

释放QHYCCD SDK的内部资源，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为ReleaseQHYCCDResource。

**函数原型：**

QHYCCD\_API uint32\_t C\_ReleaseQHYCCDResource();

**示例代码：**

int ret = C\_ReleaseQHYCCDResource();

if(ret == 0)

{

printf(“C\_ReleaseQHYCCDResource success\n”);

}

## C\_ScanQHYCCD

**函数说明：**

扫描连接到主机上的相机设备数量，函数执行成功时会返回相机数量，失败时返回-1，调用的QHYCCD SDK函数为ScanQHYCCD。

**函数原型：**

QHYCCD\_API uint32\_t C\_ScanQHYCCD();

**示例代码：**

int number = C\_ScanQHYCCD();

if(number != 0)

{

printf(“C\_ScanQHYCCD success,camera number = %d\n”, number);

}

## C\_GetQHYCCDId

**函数说明：**

获取相机的ID，相机ID包含相机型号和设备识别序列号，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDId。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDId(

uint32\_t index,

char\* id

);

**参数说明：**

index：相机的索引，从0开始计数。

id：用来返回相机ID。

**示例代码：**

char id[40] = { 0 };

int ret = C\_GetQHYCCDId(0, id);

if(ret == 0)

{

printf(“C\_GetQHYCCDId success, camera ID = %s\n”, id);

}

## C\_OpenQHYCCD

**函数说明：**

打开相机设备并返回设备句柄，执行成功时返回设备句柄，失败时返回空指针，调用的QHYCCD SDK函数为OpenQHYCCD。

**函数原型：**

QHYCCD\_API qhyccd\_handle \* C\_OpenQHYCCD(

char\* id

);

**参数说明：**

id：由C\_GetQHYCCDId函数获取的相机ID。

**示例代码：**

qhyccd\_handle\* handle = C\_OpenQHYCCD(id);

if(handle != NULL)

{

printf(“C\_OpenQHYCCD success, camera handle = 0x%x\n”, handle);

}

## C\_CloseQHYCCD

**函数说明：**

关闭相机，执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为CloseQHYCCD。

**函数原型：**

QHYCCD\_API uint32\_t C\_CloseQHYCCD(

qhyccd\_handle\* handle

);

**参数说明：**

handle：相机的设备句柄。

**示例代码：**

int ret = C\_CloseQHYCCD(handle);

if(ret == 0)

{

printf(“C\_CloseQHYCCD success\n”);

}

## C\_GetQHYCCDReadModeNumber

**函数说明：**

获取相机所支持的读出模式数量，所有相机都至少支持一种读出模式，执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDNumberOfReadModes。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDReadModeNumber(

qhyccd\_handle\* handle,

uint32\_t\* number

);

**参数说明：**

handle：设备句柄。

number：用来返回读出模式数量。

**示例代码：**

uint32\_t number = 0;

int ret = C\_GetQHYCCDReadModeNumber(handle, &number);

if(ret == 0)

{

printf(“C\_GetQHYCCDReadModeNumber success, read mode number = %d\n”, number);

}

## C\_GetQHYCCDReadModeName

**函数说明：**

获取某个读出模式的名称，执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDReadModeName函数。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDReadModeName(

qhyccd\_handle\* handle,

uint32\_t modeIndex,

char\* name

);

**参数说明：**

handle：设备句柄。

modeIndex：读出模式索引，从0开始计数。

**示例代码：**

char name[40] = { 0 };

for(int i = 0; i < number; i++)

{

int ret = C\_GetQHYCCDReadModeName(handle, 0, name);

if(ret == 0)

{

printf(“C\_GetQHYCCDReadModeName success, index = %d read mode name = %s\n”, i, name);

}

}

## C\_SetQHYCCDReadMode

**函数说明：**

设置读出模式，不同读出模式下相机会有不同特性，执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDReadMode。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDReadMode(

qhyccd\_handle\* handle,

uint32\_t modeIndex

);

**参数说明：**

handle：设备句柄。

modeIndex：读出模式索引，从0开始计数。

**示例代码：**

int ret = C\_SetQHYCCDReadMode(handle, 0);

if(ret == 0)

{

printf(“C\_SetQHYCCDReadMode success\n”);

}

## C\_SetQHYCCDStreamMode

**函数说明：**

设置单帧或连续数据流模式，执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDStreamMode函数。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDStreamMode(

qhyccd\_handle\* handle,

uint32\_t readmode

);

**参数说明：**

handle：设备句柄。

readmode：设置0为单帧模式，1为连续模式。

**示例代码：**

int ret = C\_SetQHYCCDStreamMode(handle, 0);

if(ret == 0)

{

printf(“C\_SetHYCCDStreamMode success\n”);

}

## C\_InitQHYCCD

**函数说明：**

初始化相机状态和参数，此函数会初始化相机寄存器状态并清除之前的参数设置，如曝光时间、增益等，执行成功时返回值为0，失败时返回值为-1，调用的QHYCCD SDK函数为InitQHYCCD。

**函数原型：**

QHYCCD\_API uint32\_t C\_InitQHYCCD(

qhyccd\_handle \*handle

);

**参数说明：**

handle：设备句柄。

**示例代码：**

int ret = C\_InitQHYCCD(handle);

if(ret == 0)

{

printf(“C\_InitQHYCCD success\n”);

}

## C\_SetQHYCCDDebayerOnOff

**函数说明：**

设置相机debayer开启或关闭，执行成功时返回0，失败时返回-1，仅彩色相机支持此设置，黑白相机调用此函数默认返回-1，调用的QHYCCD SDK函数为SetQHYCCDDebayerOnOff。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDDebayerOnOff(

qhyccd\_handle\* handle,

bool onoff

);

**参数说明：**

handle：设备句柄。

onoff：设置true为开启debayer，false为关闭。

**示例代码：**

int ret = C\_SetQHYCCDDebayerOnOff(handle, false);

if(ret == 0)

{

printf(“C\_SetQHYCCDDebayerOnOff success\n”);

}

## C\_SetQHYCCDParam\_Bits

**函数说明：**

设置相机图像位数，和C\_SetQHYCCDDebayerOnOff函数配合可以设置相机图像的数据格式，如RAW8/MONO8、RAW16/MONO16、RGB24，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_TRANSFERBIT, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_Bits(

qhyccd\_handle\* handle,

double bits

);

**参数说明：**

handle：设备句柄。

bits：图像位数，设置8为八位图像，16为十六位图像。

**示例代码：**

int ret = C\_SetQHYCCDParam\_Bits(handle, 16.0);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_Bits success\n”);

}

## C\_SetQHYCCDBinMode

**函数说明：**

设置图像合并模式，如1x1 bin、2x2 bin，执行成功时返回0，失败时返回-1，另外此合并为软件合并，不会减少图像传输的数据量，也不会增加连续模式下的帧率，通常需要和C\_SetQHYCCDResolution函数配合使用，调用的QHYCCD SDK函数为SetQHYCCDBinMode。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDBinMode(

qhyccd\_handle\* handle,

uint32\_t wbin,

uint32\_t hbin

);

**参数说明：**

handle：设备句柄。

wbin：水平方向的合并倍数，设置1时为不合并，2时为2倍合并。

hbin：垂直方向的合并倍数，设置1时为不合并，2时为2倍合并。

**示例代码：**

int ret = C\_SetQHYCCDBinMode(handle, 1, 1);

if(ret == 0)

{

printf(“C\_SetQHYCCDBinMode success\n”);

//imagew imageh can get from C\_GetQHYCCDChipInfo

ret = C\_SetQHYCCDResolution(handle, 0, 0, iamgew, imageh);

if(ret == 0)

{

printf(“C\_SetQHYCCDResolution success\n”);

}

}

## C\_SetQHYCCDResolution

**函数说明：**

设置图像ROI或和C\_SetQHYCCDBinMode配合使用来设置图像分辨率，执行成功时返回0，失败时返回-1，另外，若相机支持硬件ROI，此时设置ROI可以减少数据传输量，增加连续模式帧率，调用的QHYCCD SDK函数为SetQHYCCDResolution。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDResolution(

qhyccd\_handle\* handle,

uint32\_t x,

uint32\_t y,

uint32\_t xsize,

uint32\_t ysize

);

**参数说明：**

handle：设备句柄。

x：图像起始位置的x左边，以左上角为参考点。

y：图像起始位置的y左边，以左上角为参考点。

xsize：ROI区域的宽度。

ysize：ROI区域的高度。

**示例代码：**

int ret = C\_SetQHYCCDResolution(handle, 100, 100, 500, 500);

if(ret == 0)

{

printf(“C\_SetQHYCCDResolution success\n”);

}

## C\_GetQHYCCDChipInfo

**函数说明：**

获取相机芯片的硬件信息，包括芯片尺寸，图像尺寸，像素尺寸，图像位数，执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDChipInfo。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDChipInfo(

qhyccd\_handle\* handle,

double\* chipw,

double\* chiph,

uint32\_t\* imagew,

uint32\_t\* imageh,

double\* pixelw,

double\* pixelh,

uint32\_t\* bpp

);

**参数说明：**

handle：设备句柄。

chipw：芯片的物理宽度，单位为毫米。

chiph：芯片的物理高度，单位为毫米。

imagew：图像宽度，单位为像素。

imageh：图像高度，单位为像素。

pixel：像素的物理宽度，单位为微米。

pixelh：像素的物理高度，单位为微米。

bpp：图像数据位数，会根据位数设置而产生变化。

**示例代码：**

double chipw = 0.0, chiph = 0.0, pixelw = 0.0, pixelh = 0.0;

uint32\_t imagew = 0, imageh = 0, bpp = 0;

int ret = C\_GetQHYCCDChipInfo(handle, &chipw, &chiph, &imagew, &imageh, &pixelw, &pixelh, &bpp);

if(ret == 0)

{

printf(“C\_GetQHYCDChipInfo success,chip size %fmm x %fmm,image size %d x %d,pixel size %f

um x %fum,bpp = %d\n”, chipw, chiph, imagew, imageh, pixelw, pixelh, bpp);

}

## C\_GetQHYCCDEffectiveArea

**函数说明：**

获取图像有效区域的位置及尺寸信息，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDEffectiveArea。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDEffectiveArea(

qhyccd\_handle\* handle,

uint32\_t\* startX,

uint32\_t\* startY,

uint32\_t\* sizeX,

uint32\_t\* sizeY

);

**参数说明：**

handle：设备句柄。

startX：有效区域起始位置的x坐标，以图像左上角为参考点。

startY：有效区域起始位置的y坐标，以图像左上角为参考点。

sizeX：图像有效区域的宽度。

sizeY：图像有效区域的高度。

**示例代码：**

uint32\_t startX = 0, startY = 0, sizeX = %d sizeY = 0;

int ret = C\_GetQHYCCDEffectiveArea(handle, startX, startY, sizeX, sizeY);

if(ret == 0)

{

printf(“C\_GetQHYCCDEffectiveArea success, effective area start position (%d, %d),size %d x %d

\n”, startX, startY, sizeX, sizeY);

}

## C\_GetQHYCCDOverScanArea

**函数说明：**

获取图像过扫区的位置及尺寸信息，功能等同于QHYCCD SDK的GetQHYCCDOverScanArea函数，函数执行成功时返回0，失败时返回-1。正常情况下，过扫区位于图像边缘，且位置固定不变，不同型号相机过扫区位置不一定相同，同一个相机可能有多个过扫区，但此函数只会返回其中一个过扫区的位置和尺寸信息。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDOverScanArea(

qhyccd\_handle\* handle,

uint32\_t\* startX,

uint32\_t\* startY,

uint32\_t\* sizeX,

uint32\_t\* sizeY

);

**参数说明：**

handle：设备句柄。

startX：过扫区起始位置的x坐标，以图像左上角为参考点。

startY：过扫区起始位置的y坐标，以图像左上角为参考点。

sizeX：图像过扫区的宽度。

sizeY：图像过扫区的高度。

**示例代码：**

uint32\_t startX = 0, startY = 0, sizeX = %d sizeY = 0;

int ret = C\_GetQHYCCDOverScanArea(handle, startX, startY, sizeX, sizeY);

if(ret == 0)

{

printf(“C\_GetQHYCCDOverScanArea success, overscan area start position (%d, %d),size %d x %d

\n”, startX, startY, sizeX, sizeY);

}

## C\_GetQHYCCDMemLength

**函数说明：**

获取相机图像数据内存长度，函数获取的数据长度会比实际的多，计算公式为(imagw+100) \* (imageh+100) \* 4，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDMemoryLength。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDMemLength(

qhyccd\_handle\* handle

);

**参数说明：**

handle：设备句柄。

**示例代码：**

int length = C\_GetQHYCCDMemLength(handle);

if(ret == 0)

{

printf(“C\_GetQHYCCDMemLength success\n”);

}

## C\_SetQHYCCDParam\_Exposure

**函数说明：**

设置相机拍摄时的曝光时间，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_EXPOSURE, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_Exposure(

qhyccd\_handle\* handle,

double time

);

**参数说明：**

handle：设备句柄。

time：曝光时间参数值，单位为微秒，例如设置50000.0为50ms。

**示例代码：**

int ret = C\_SetQHYCCDParam(handle, CONTROL\_EXPOSURE, 50000.0);

if(ret == 0)

{

printf(“C\_SetQHCCDParam\_Exposure success\n”);

}

## C\_SetQHYCCDParam\_Gain

**函数说明：**

设置相机增益，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_GAIN, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_Gain(

qhyccd\_handle\* handle,

double gain

);

**参数说明：**

handle：设备句柄。

gain：增益参数值。

**示例代码：**

int ret = C\_SetQHYCCDParam\_Gain(handle, 20);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_Gain success\n”);

}

## C\_SetQHYCCDParam\_Offset

**函数说明：**

设置相机offset，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_OFFSET, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_Offset(

qhyccd\_handle\* handle,

double offset

);

**参数说明：**

handle：设备句柄。

offset：offset参数值。

**示例代码：**

int ret = C\_SetQHYCCDParam\_Offset(handle, 20);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_Offset success\n”);

}

## C\_SetQHYCCDParam\_Traffic

**函数说明：**

设置相机traffic，traffic值越小帧率越高，设置0时为最高，需要注意的是此函数无法使软件中显示的FPS参数固定为某一值，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_USBTRAFFIC, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_Traffic(

qhyccd\_handle\* handle,

double traffic

);

**参数说明：**

handle：设备句柄。

traffic：traffic参数值。

**示例代码：**

int ret = C\_SetQHYCCDParam\_Traffic(handle, 0);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_Traffic success\n”);

}

## C\_SetQHYCCDParam\_WBR

**函数说明：**

设置彩色相机R通道增益，仅彩色型号相机支持此设置，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_WBR, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_WBR(

qhyccd\_handle\* handle,

double wbr

);

**参数说明：**

handle：设备句柄。

wbr：R通道增益参数值。

**示例代码：**

int ret = C\_SetQHYCCDParam\_WBR(handle, 20);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_WBR success\n”);

}

## C\_SetQHYCCDParam\_WBG

**函数说明：**

设置彩色相机G通道增益，仅彩色型号相机支持此设置，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_WBG, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_WBG(

qhyccd\_handle\* handle,

double wbg

);

**参数说明：**

handle：设备句柄。

wbg：G通道增益参数值。

**示例代码：**

int ret = C\_SetQHYCCDParam\_WBG(handle, 20);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_WBG success\n”);

}

## C\_SetQHYCCDParam\_WBB

**函数说明：**

设置彩色相机B通道增益，仅彩色型号相机支持此设置，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_WBB, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_WBB(

qhyccd\_handle\* handle,

double wbb

);

**参数说明：**

handle：设备句柄。

wbb：B通道增益参数值。

**示例代码：**

int ret = C\_SetQHYCCDParam\_WBB(handle, 20);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_WBB success\n”);

}

## C\_SetQHYCCDParam\_Brightness

**函数说明：**

设置图像亮度，设置值越大图像越亮，反之越暗，参数设置范围固定为-1.0~1.0，参数设置步长固定为0.1，默认参数值为0.0，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_BRIGHTNESS, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_Brightness(

qhyccd\_handle\* handle,

double brightness

);

**参数说明：**

handle：设备句柄。

brightness：亮度参数值。

**示例代码：**

int ret = C\_SetQHYCCDParam\_Brightness(handle, 0.0);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_Brightness success\n”);

}

## C\_SetQHYCCDParam\_Contrast

**函数说明：**

设置图像对比度，设置值越大，对比度越高，反之越低，参数设置范围固定为-1.0~1.0，参数设置步长固定为0.1，默认参数值为0.0，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_CONTRAST, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_Contrast(

qhyccd\_handle\* handle,

double contrast

);

**参数说明：**

handle：设备句柄。

contrast：对比度参数值。

**示例代码：**

int ret = C\_SetQHYCCDParam\_Contrast(handle, 0.0);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_Contrast success\n”);

}

## C\_SetQHYCCDParam\_Gamma

**函数说明：**

设置图像gamma值，设置值越大图像越亮，反之越暗，参数设置范围固定为-1.0~1.0，参数设置步长固定为0.1，默认参数值为1.0，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDParam(handle, CONTROL\_GAMMA, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_Gamma(

qhyccd\_handle\* handle,

double gamma

);

**参数说明：**

handle：设备句柄。

gamma：gamma参数值。

**示例代码：**

int ret = C\_SetQHYCCDParam\_Gamma(handle, 1.0);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_Traffic success\n”);

}

## C\_ExpQHYCCDSingleFrame

**函数说明：**

控制相机开始单帧模式曝光，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为ExpQHYCCDSingleFrame。

**函数原型：**

QHYCCD\_API uint32\_t C\_ExpQHYCCDSingleFrame(

qhyccd\_handle\* handle

);

**参数说明：**

handle：设备句柄。

**示例代码：**

int ret = C\_ExpQHYCCDSingleFrame(handle);

if(ret == 0)

{

printf(“C\_ExpQHYCCDSingleFrame success\n”);

}

## C\_GetQHYCCDSingleFrame

**函数说明：**

获取单帧模式下的图像数据，函数为阻塞运行，长曝光模式时软件卡顿为正常现象，拍摄完成后会固定按照从左到右从上到下的顺序读取芯片像素数据，并依次存放到一维数组中，单帧模式数据格式通常固定为黑白十六位模式，此时两个数组元素存储一个像素的数据，偶数索引数组元素存储低位数据，奇数索引数组元素存储高位数据，原始像素值计算方式为高位数据\*256+高位数据，例如ImgData[0]、ImgData[1]存储第一个像素的数据，像素值为ImgData[1]\*256+ImgData[0]，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数包括GetQHYCCDSingleFrame。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDSingleFrame(

qhyccd\_handle\* handle,

uint32\_t\* w,

uint32\_t\* h,

uint32\_t\* bpp,

uint32\_t\* channels,

uint8\_t\* imgdata

);

**参数说明：**

handle：设备句柄。

w：返回获取到的图像宽度。

h：返回获取到的图像高度。

bpp：返回获取到的图像位数，单帧模式通常为16。

channels：返回获取到的图像通道数，黑白图像固定为1，彩色图像固定为3。

imgdata：返回获取到的图像数据。

**示例代码：**

uint32\_t w = 0, h = 0, bpp = 0, channels = 0;

uint8\_t\* imgdata = (uint8\_t\*)malloc(imagew \* imageh \* 2 \* sizeof(uint8\_t));//for RAW16/MINI16

int ret = C\_GetQHYCCDSingleFrame(handle, &w, &h, \*bpp, &channels, imgdata);

if(ret == 0)

{

printf(“C\_GetQHYCCDSingleFrame success,w = %d h = %d bpp = %d channels = %d\n”,

w, h, bpp, channels);

}

## C\_CancelQHYCCDExposingAndReadout

**函数说明：**

取消相机正在执行的单帧拍摄操作且不会读出图像数据，执行成功时返回0，失败时返回1，调用的QHYCCD SDK函数为CancelQHYCCDExposingAndReadout。

**函数原型：**

QHYCCD\_API uint32\_t C\_CancelQHYCCDExposingAndReadout(

qhyccd\_handle\* handle

);

**参数说明：**

handle：设备句柄。

**示例代码：**

int ret = C\_CancelQHYCCDExposingAndReadout(handle);

if(ret == 0)

{

printf(“C\_CancelQHYCCDExposingAndReadout success\n”);

}

## C\_BeginQHYCCDLive

**函数说明：**

开始连续模式拍摄，调用此函数后QHYCCD SDK内部线程会自动获取图像数据，执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为BeginQHYCCDLive。

**函数原型：**

QHYCCD\_API uint32\_t C\_BeginQHYCCDLive(

qhyccd\_handle\* handle

);

**参数说明：**

handle：设备句柄。

**示例代码：**

int ret = C\_BeginQHYCCDLive(handle);

if(ret == 0)

{

printf(“C\_BeginQHYCCDLive success\n”);

}

## C\_GetQHYCCDLiveFrame

**函数说明：**

从QHYCCD SDK线程中获取一帧图像数据，函数运行模式为轮询模式，获取到图像数据时返回0，未获取到图像数据时返回-1，通常情况下会多次执行函数才能获取一帧图像数据，拍摄完成后会固定按照从左到右从上到下的顺序读取芯片像素数据，并依次存放到一维数组中，黑白八位模式下一个数组元素存储一个像素的数据，黑白十六位模式下两个数组元素存储一个像素的数据，偶数索引数组元素存储低位数据，奇数索引数组元素存储高位数据，原始像素值计算方式为高位数据\*256+高位数据，例如ImgData[0]、ImgData[1]存储第一个像素的数据，像素值为ImgData[1]\*256+ImgData[0]，八位彩色模式下三个数组元素存储一个像素数据，分别存储B、G、R三个通道的图像数据，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDLiveFrame。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDLiveFrame(

qhyccd\_handle\* handle,

uint32\_t\* w,

uint32\_t\* h,

uint32\_t\* bpp,

uint32\_t\* channels,

uint8\_t\* imgdata

);

**参数说明：**

handle：设备句柄。

w：返回获取到的图像宽度。

h：返回获取到的图像高度。

bpp：返回获取到的图像位数。

channels：返回获取到的图像通道数，黑白图像固定为1，彩色图像固定为3。

imgdata：返回获取到的图像数据。

**示例代码：**

uint32\_t w = 0, h = 0, bpp = 0, channels = 0;

uint8\_t\* imgdata = (uint8\_t\*)malloc(imagew \* imageh \* 2 \* sizeof(uint8\_t));//for RAW16/MINI16

int ret = C\_GetQHYCCDLiveFrame(handle, &w, &h, &bpp, &channels, imgdata);

if(ret == 0)

{

printf(“C\_GetQHYCCDLiveFrame success,w = %d h = %d bpp = %d channels = %d\n”,

w h, bpp, channels);

}

## C\_StopQHYCCDLive

**函数说明：**

停止连续模式拍摄，结束QHYCCD SDK内部线程，执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为StopQHYCCDLive。

**函数原型：**

QHYCCD\_API uint32\_t C\_StopQHYCCDLive(

qhyccd\_handle\* handle

);

**参数说明：**

handle：设备句柄。

**示例代码：**

int ret = C\_StopQHYCCDLive(handle);

if(ret == 0)

{

printf(“C\_StopQHYCCDLive success\n”);

}

## C\_GetQHYCCDTrigerInterfaceNumber

**函数说明：**

获取相机支持的硬件触发接口数量，部分相机带有多个硬件触发接口，如SMA或GPIO，可以根据需求进行选择，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDTrigerInterfaceNumber。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDTrigerInterfaceNumber(

qhyccd\_handle\* handle,

uint32\_t\* interfaceNumber

);

**参数说明：**

handle：设备句柄。

interfaceNumber：返回触发接口数量。

**示例代码：**

uint32\_t number = 0;

int ret = C\_GetQHYCCDTrigerInterfaceNumber(handle, &number);

if(ret == 0)

{

printf(“C\_GetQHYCCDTrigerInterfaceNumber success,interface number = %d\n”, number);

}

## C\_GetQHYCCDTrigerInterfaceName

**函数说明：**

获取硬件触发接口的名称，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDTrigerInterfaceName。

**函数原型：**

QHYCCD\_API uint32\_t C\_GetQHYCCDTrigerInterfaceName(

qhyccd\_handle\* handle,

uint32\_t interfaceIndex,

char\* name

);

**参数说明：**

handle：设备句柄。

interfaceIndex：触发接口索引号。

name：返回触发接口的名称。

**示例代码：**

char name[40] = { 0 };

for(int i = 0; i < number; i++)

{

int ret = C\_GetQHYCCDTrigerInterfaceName(handle, i, name);

if(ret == 0)

{

printf(“C\_GetQHYCCDTrigerInterfaceName success i = %d name = %d\n”, i, name);

}

}

## C\_SetQHYCCDTrigerInterface

**函数说明：**

设置相机外触发功能硬件触发接口，部分相机带有多个硬件触发接口，如SMA或GPIO，此时需要人为设置要使用的触发接口，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDTrigerInterface。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDTrigerInterface(

qhyccd\_handle\* handle,

uint32\_t interfaceIndex

);

**参数说明：**

handle：设备句柄。

interfaceIndex：触发接口的索引号。

**示例代码：**

int ret = C\_SetQHYCCDTrigerInterface(handle, 0);

if(ret == 0)

{

printf(“C\_SetQHYCCDTrigerInterface success\n”);

}

## C\_SetQHYCCDTrigerFunction

**函数说明：**

设置触发功能开启和关闭，开启触发功能时，相机将受硬件触发信号控制进行拍摄，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDTrigerFunction。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDTrigerFunction(

qhyccd\_handle\* handle,

bool onoff

);

**参数说明：**

handle：设备句柄。

onoff：设置true为开启外触发功能，false为关闭外触发功能。

**示例代码：**

int ret = C\_SetQHYCCDTrigerFunction(handle, true);

if(ret == 0)

{

printf(“C\_SetQHYCCDTrigerFunction success\n”);

}

## C\_EnableQHYCCDTrigerOut

**函数说明：**

使能触发输出功能，此函数可以设置只启用触发输出功能，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为EnableQHYCCDTrigerOut。

**函数原型：**

QHYCCD\_API uint32\_t C\_EnableQHYCCDTrigerOut(

qhyccd\_handle\* handle3

);

**参数说明：**

handle：设备句柄。

**示例代码：**

int ret = C\_EnableQHYCCDTrigerOut(handle);

if(ret == 0)

{

printf(“C\_EnableQHYCCDTrigerOut success\n”);

}

## C\_EnableQHYCCDBurstMode

**函数说明：**

设置burst模式开启和关闭，此模式仅在连续模式下可以开启，开启Burst模式后相机进入IDLE状态，此状态下可以通过发送软件指令控制相机拍摄指定数量的图像，此功能可以代替单帧模式更快速地获取图像数据，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为EnableQHYCCDBurstMode。

**函数原型：**

QHYCCD\_API uint32\_t C\_EnableQHYCCDBurstMode(

qhyccd\_handle\* handle,

bool onoff

);

**参数说明：**

handle：设备句柄。

onoff：设置true为开启burst模式，false为关闭burst模式。

**示例代码：**

int ret = C\_EnableQHYCCDBurstMode(handle, true);

if(ret == 0)

{

printf(“C\_EnableQHYCCDBurstMode success\n”);

}

## C\_SetQHYCCDBurstModeStartEnd

**函数说明：**

设置Burst模式图像开始和结束帧序号，拍摄时相机将输出开始和结束帧中间的图像，例如设置start、end分别为1、3，拍摄时相机将输出帧序号为2的图像，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDBurstStartEnd。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDBurstModeStartEnd(

qhyccd\_handle\* handle,

uint32\_t start,

uint32\_t end

);

**参数说明：**

handle：设备句柄。

start：起始帧的序号。

end：结束帧的序号。

**示例代码：**

int ret = C\_SetQHYCCDBurstModeStartEnd(handle, 1, 3);

if(ret == 0)

{

printf(“C\_SetQHYCCDBurstModeStartEnd success\n”);

}

## C\_SetQHYCCDBurstModePatchNumber

**函数说明：**

设置Burst模式补充数据包的长度，Burst模式下若相机缓存中数据过少，有可能会导致图像数据无法完整读出，此时可以通过此函数使图像数据可以被正常读出，通常设置32001以内的数值，函数执行成功时返回0，执行失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDBurstModePatchNumber。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDBurstModePatchNumber(

qhyccd\_handle\* handle,

uint32\_t number

);

**参数说明：**

handle：设备句柄。

number：补包的长度，通常为32001以内。

**示例代码：**

int ret = C\_SetQHYCCDBurstModePatchNumber(handle, 32001);

if(ret == 0)

{

printf(“C\_SetQHYCCDBurstModePatchNumber success\n”);

}

## C\_SetQHYCCDBurstIDLE

**函数说明：**

Burst模式下调用此函数可以使相机处于IDLE状态，当解除IDLE状态时相机会按照设置发出指定帧数的图像数据，通常和ReleaseQHYCCDBurstIDLE函数配合使用，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为SetQHYCCDBurstIDLE。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDBurstIDLE(

qhyccd\_handle\* handle

);

**参数说明：**

handle：设备句柄。

**示例代码：**

int ret = C\_SetQHYCCDBurstIDLE(handle);

if(ret == 0)

{

printf(“C\_SetQHYCCDBurstIDLE success\n”);

}

## C\_ReleaseQHYCCDBurstIDLE

**函数说明：**

Burst模式下调用此函数可以使相机解除IDLE状态，当解除IDLE状态时相机会按照设置发出指定帧数的图像数据，通常和SetQHYCCDBurstIDLE函数配合使用，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为ReleaseQHYCCDBurstIDLE。

**函数原型：**

QHYCCD\_API uint32\_t C\_ReleaseQHYCCDBurstIDLE(

qhyccd\_handle\* handle

);

**参数说明：**

handle：设备句柄。

**示例代码：**

int ret = C\_SetQHYCCDBurstIDLE(handle);

if(ret == 0)

{

printf(“C\_SetQHYCCDBurstIDLE success\n”);

}

msleep(200);

ret = C\_ReleaseQHYCCDBurstIDLE(handle);

if(ret == 0)

{

printf(“C\_ReleaseQHYCCDBurstIDLE success\n”);

}

## C\_GetQHYCCDParam\_CurTemperature

**函数说明：**

获取相机当前温度，此函数为单次获取，若要实现显示温度变化，需持续调用函数获取温度，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK函数为GetQHYCCDParam(handle, CONTROL\_CURTEMP)。

**函数原型：**

QHYCCD\_API double C\_GetQHYCCDParam\_CurTempture(

qhyccd\_handle\* handle

);

**参数说明：**

handle：设备句柄。

**示例代码：**

double temp = C\_GetQHYCCDParam\_CurTemperature(handle);

printf(“C\_GetQHYCCDParam\_CurTemperature success,current temperature = %f\n”, temp);

## C\_SetQHYCCDParam\_TargetTemperature

**函数说明：**

设置制冷器目标温度，设置完成后相机将自动进入自动控制模式，相机内部程序自动调节制冷器功率，使相机维持目标温度，需要注意的是相机制冷能力有上限，一般制冷范围为低于环境温度30~35℃，若设置的目标温度与环境温度相差过大，会超出相机制冷能力范围，导致相机温度无法到达目标温度，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK为SetQHYCCDParam(handle, CONTROL\_COOLER, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_TargetTemperature(

qhyccd\_handle\* handle,

double temp

);

**参数说明：**

handle：设备句柄。

temp：目标温度。

**示例代码：**

int ret = C\_SetQHYCCDParam\_TargetTemperature(handle, 0.0);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_TargetTemperature success\n”);

}

## C\_SetQHYCCDParam\_CoolerPWM

**函数说明：**

设置制冷器的制冷功率，调用此函数可以使相机自动进入手动控制模式，制冷器将以固定功率工作，当功率设置为0时为关闭相机制冷，函数执行成功时返回0，失败时返回-1，调用的QHYCCD SDK为SetQHYCCDParam(handle, CONTROL\_MANULPWM, value)。

**函数原型：**

QHYCCD\_API uint32\_t C\_SetQHYCCDParam\_CoolerPWM(

qhyccd\_handle\* handle,

double pwm

);

**参数说明：**

handle：设备句柄。

pwm：制冷器功率，设置范围为0~255，设置步长为1。

**示例代码：**

int ret = C\_SetQHYCCDParam\_CoolerPWM(handle, 255.0);

if(ret == 0)

{

printf(“C\_SetQHYCCDParam\_CoolerPWM success\n”);

}