



QHYCCD C WRAPPER API

说明文档

V1.0

目录

1. 项目介绍	1
2. 函数介绍	2
2.1. C_InitQHYCCDResource	2
2.2. C_ReleaseQHYCCDResource	2
2.3. C_ScanQHYCCD	2
2.4. C_GetQHYCCDId	3
2.5. C_OpenQHYCCD	3
2.6. C_CloseQHYCCD	4
2.7. C_GetQHYCCDReadModeNumber	4
2.8. C_GetQHYCCDReadModeName	4
2.9. C_SetQHYCCDReadMode	5
2.10. C_SetQHYCCDStreamMode	5
2.11. C_InitQHYCCD	6
2.12. C_SetQHYCCDDebayerOnOff	6
2.13. C_SetQHYCCDParam_Bits	7
2.14. C_SetQHYCCDBinMode	7
2.15. C_SetQHYCCDResolution	8
2.16. C_SetQHYCCDChipInfo	8
2.17. C_GetQHYCCDEffectiveArea	9
2.18. C_GetQHYCCDOverScanArea	10
2.19. C_GetQHYCCDMemLength	11
2.20. C_SetQHYCCDParam_Exposure	11
2.21. C_SetQHYCCDParam_Gain	12
2.22. C_SetQHYCCDParam_Offset	12
2.23. C_SetQHYCCDParam_Traffic	12
2.24. C_SetQHYCCDParam_WBR	13
2.25. C_SetQHYCCDParam_WBG	13
2.26. C_SetQHYCCDParam_WBB	14
2.27. C_SetQHYCCDParam_Brightness	14
2.28. C_SetQHYCCDParam_Contrast	15
2.29. C_SetQHYCCDParam_Gamma	15
2.30. C_ExpQHYCCDSingleFrame	16
2.31. C_GetQHYCCDSingleFrame	16
2.32. C_CancelQHYCCDExposingAndReadout	17
2.33. C_BeginQHYCCDLive	17
2.34. C_GetQHYCCDLiveFrame	18
2.35. C_StopQHYCCDLive	19
2.36. C_GetQHYCCDTrigerInterfaceNumber	19
2.37. C_GetQHYCCDTrigerInterfaceName	20
2.38. C_SetQHYCCDTrigerInterface	20



2.39. C_SetQHYCCDTrigerFunction	21
2.40. C_EnableQHYCCDTrigerOut	21
2.41. C_EnableQHYCCDBurstMode	21
2.42. C_SetQHYCCDBurstModeStartEnd	22
2.43. C_SetQHYCCDBurstModePatchNumber	23
2.44. C_SetQHYCCDBurstIDLE	23
2.45. C_ReleaseQHYCCDBurstIDLE	23
2.46. C_GetQHYCCDParam_CurTemperature	24
2.47. C_SetQHYCCDParam_TargetTemperature	24
2.48. C_SetQHYCCDParam_CoolerPWM	25

1. 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/。

SDK API手册

记录SDK API说明的手册，包含常用函数的使用方式及一些简单的示例代码。

[中文版下载地址](#)

[英文版下载地址](#)

2. 函数介绍

注：QHYCCD_API 定义为 __declspec(dllexport)。

2.1. 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");
}
```

2.2. 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");
}
```

2.3. 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);  
}
```

2.4. 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);  
}
```

2.5. 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);  
}
```

2.6. 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");  
}
```

2.7. 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);  
}
```

2.8. 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);  
    }  
}
```

2.9. 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");  
}
```

2.10. 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");  
}
```

2.11. 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");  
}
```

2.12. 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");
}
```

2.13. 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");
}
```

2.14. 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, imagew, imageh);
    if(ret == 0)
    {
        printf("C_SetQHYCCDResolution success\n");
    }
}
```

2.15. 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");
}
```

2.16. 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。

pixelw：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_GetQHYCCDChipInfo 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);  
}
```

2.17. 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);  
}
```

2.18. 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);  
}
```

2.19. 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(\text{imagw}+100) * (\text{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");  
}
```

2.20. 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");  
}
```

2.21. 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");  
}
```

2.22. 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");  
}
```

2.23. 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");  
}
```

2.24. 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");  
}
```

2.25. 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");  
}
```

2.26. 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");  
}
```

2.27. 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");
}
```

2.28. 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");
}
```

2.29. 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");
}
```

2.30. 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");
}
```

2.31. 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，并依次存放成一维数组中，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);  
}
```

2.32. 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");  
}
```

2.33. 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");  
}
```

2.34. 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);
}
```

2.35. 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");
}
```

2.36. 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);
}
```

2.37. 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);  
    }  
}
```

2.38. 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");  
}
```

2.39. 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");  
}
```

2.40. 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");  
}
```

2.41. 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");  
}
```

2.42. 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");  
}
```

2.43. 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，有可能会造成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");  
}
```

2.44. 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");  
}
```

2.45. 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");  
}
```

2.46. 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_CurTempature(  
    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);
```

2.47. 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");  
}
```

2.48. 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");  
}
```