QHYCCD SDK 说明文档

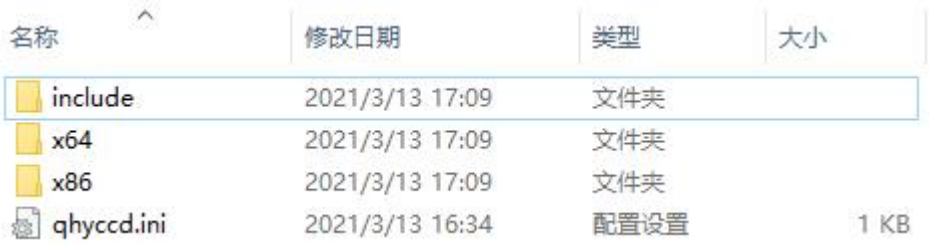
一、SDK 介绍

QHYCCD SDK 是由 Visual Studio 编译生成的 C++ 链接库文件，链接库里面集成了所有用于控制相机的函数。

通过 SDK 可以进行二次开发和软件更新，当进行二次开发时，需要根据编译语言的不同采用不同的调用方式。另外，使用时需要注意软件的位数，SDK 的位数只取决于软件的位数，32 位软件使用 32 位的 SDK，64 位软件则使用 64 位的 SDK。

SDK 可以从官网上下载，下载位置：https://www.qhyccd.cn/html/prepub/log.html#!log.md

页面里记录了当前最新版本 SDK。下载之后是一个压缩包， include 文件夹中的是 SDK 的头文件，x64 文件夹中的是 64 位的 SDK，x86 文件夹中包含的是 32 位 SDK，ini 是配置文件，里面包含了一些功能的配置参数。



除了 qhyccd.dll 外，压缩包中还包含了几个额外的文件，分别是 qhyccd.exp、qhyccd.lib、ftd2xx.dll、tbb.dll、winusb.dll、 msvcp90.dll 和 msvcr90.dll。 qhyccd.exp 是 SDK 的导出库文件，多数时候用不到，可以忽略；qhyccd.lib 是静态链接库，当使用静态链接方式进行开发时需要使用这个库文件；ftd2xx.dll 是 ftd 芯片的库，多数时候用不到，可以忽略；tbb.dll 是做 C#开发时用到的库，有需要的话可以和 qhyccd.dll 放在同一起目录下。msvcr90.dll 和 msvcp90.dll 是 VC++运行库，当程序提示缺少运行库时可以拷贝到程序目录下。

二、函数解析

SDK 中大部分 API 函数都会通过返回值来判断函数是否执行成功，这个返回值是事先定义在 qhyccderr.h 中的宏，通常返回值是 QHYCCD_SUCCESS 或 QHYCCD_ERROR，其值分别为 0 和-1。个别函数没有返回值或

是其他类型的返回值，这些函数会在后面做特殊说明。

1.uint32_t InitQHYCCDResource(void);

函数说明：

初始化 SDK 的资源，若函数执行成功，则返回 QHYCCD_SUCCESS。在程序开始时调用一次即可，不要多次调用，多次调用可能会导致程序崩溃。

示例代码：

```
uint32_t ret = QHYCCD_ERROR;
ret = InitQHYCCDResource();
if(ret == QHYCCD_SUCCESS)
{
	printf("Initialize QHYCCD resource success.\n");
}
else
{
	printf("Initialize QHYCCD resource fail.\n");
}
```

2.uint32_t ScanQHYCCD(void);

函数说明：

扫描已连接的 QHYCCD 相机，执行完成后会返回扫描到的设备数量。

示例代码：

```
int num = 0;
num = ScanQHYCCD();
if(num > 0)
{
	printf("%d cameras has been connected\n",num);
}
else
{
	printf("no camera has been connected\n");
}
```

3.uint32_t GetQHYCCDId(uint32_t index,char *id);

参数介绍：

index：QHYCCD 设备列表中设备的索引值，其取值范围根据相机的数量的返回值决定；

id：用来存储相机 ID 的变量；

函数说明：

获取相机的 ID，获取到的 ID 会存放到字符数组中，此 ID 可以用来打开相机设备，获取相机句柄，函数执行成功成功返回 QHYCCD_SUCCESS。每个相机的 ID 都由相机型号和序列号组成。如 QHY183C-c915484fa76ea7552，前面的 QHY183C 是相机型号，后面的 c915484fa76ea7552 是相机的序列号。每个相机都有其独有的序列号，即使是相同型号的相机也会使用不同的序列号，且序列号在正常情况下是固定不变。

示例代码：

```
int i,ret;
char id[32] = {0};
for(i = 0;i < camNum;i++) //camNum 为 ScanQHYCCD 的返回值
{
	ret = GetQHYCCDId(i,id);
	if(ret == QHYCCD_SUCCESS)
	{
		printf("Found connected camera,the id is %s\n",id);
	}
	else
	{
		printf("some errors occered!(%d %d)\n",i,ret);
	}
}
```

4.uint32_t OpenQHYCCD(char *id);

参数说明：

id：相机的 ID，可以通过 GetQHYCCDId 函数获取；

函数说明：

通过相机的 ID 来打开相机，成功后返回相机的设备句柄，通过此句柄可以控制相机功能。若句柄不为空则说明函数执行成功。

示例代码：

```
qhyccd_handle *camhandle = NULL;
camhandle = OpenQHYCCD(id);
if(camhandle != NULL)
{
	printf("Open QHYCCD success!\n");
}
else
{
```

```
    printf("Open QHYCCD failed!\n");
}
```

5.uint32_t CloseQHYCCD(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄；

函数说明：

关闭相机，断开与相机的连接。成功返回 QHYCCD_SUCCESS。

示例代码：

```
int ret = QHYCCD_ERROR;
ret = CloseQHYCCD(camhandle);
if(ret == QHYCCD_SUCCESS)
{
    printf("Close camera success.\n");
}
else
{
    printf("Close camera failed.");
}
```

6.uint32_t ReleaseQHYCCDResource(void);

函数说明：

释放相机的资源， 若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int ret = QHYCCD_ERROR;
ret = ReleaseQHYCCDResource();
if(ret == QHYCCD_SUCCESS)
{
    printf("Release QHYCCD resource success.\n");
}
else
{
    printf("Release QHYCCD resource failed.\n");
}
```

7.uint32_t GetQHYCCDNumberOfReadMode(qhyccd_handle *h,uint32_t *numModes);

参数说明：

handle：相机设备的句柄；

numModes：用于存储相机读出模式数量的变量；

函数说明：

获取相机读出模式的数量，每个相机至少有一个读出模式，不同的读出模式会具有不同的名称，另外，个别相机的最大图像分辨率也会随着读出模式不同而发生改变，且不同读出模式会有不同的特性，具体可以联系我们的技术支持以了解不同相机不同读出模式的差异以及特性。函数执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t ret, numModes;
ret = GetQHYCCDNumberOfReadMode(camhandle, &numModes);
if(ret = QHYCCD_SUCCESS)
{
	printf("Set stream mode success!\n" );
}
else
{
	printf("Set stream mode success!\n" );
}
```

8.uint32_t GetQHYCCDReadModeName(qhyccd_handle *h, uint32_t modeNumber, char* name);

参数说明：

handle：相机设备的句柄；

modeNumber：相机读出模式的编号，其取值范围根据读出模式数量决定；

name：存储读出模式名称的变量；

函数说明：

获取读出模式的名称。函数执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t ret = QHYCCD_ERROR;
char name[80] = { 0 };
for(int i = 0; i < numModes; i ++)
{
	ret = GetQHYCCDReadModeName(camhandle, i, name);
	if(ret == QHYCCD_ERROR)
	{
		printf("Current read mode is %d,its name is %s.\n", i, name);
```

```
    }
}
```

9.uint32_t GetQHYCCDReadModeResolution(qhyccd_handle *h, uint32_t modeNumber, uint32_t* width, uint32_t* height);

参数说明：

handle：相机设备的句柄；

modeNumber：相机读出模式的编号，其取值范围根据读出模式数量决定；

width：用于存储某一读出模式下图像宽度的变量；

height：用于存储某一读出模式下图像高度的变量；

函数说明：

用于获取相机在不同读出模式下的图像尺寸，多数相机的图像尺寸是固定的，不过个别相机在不同读出模式下会使用不同的分辨率，如 QHY42PRO、QHY2020、QHY294PRO。函数执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t ret = QHYCCD_ERROR;
uint32_t width,height;
for(int i = 0; i < numModes; i ++)
{
    ret = GetQHYCCDReadModeResolution(camhandle, i, width, height);
    if(ret == QHYCCD_ERROR)
    {
        printf("Current read mode is %d,its resolution is %d x %d.\n", width, height);
    }
}
```

10.uint32_t SetQHYCCDReadMode(qhyccd_handle *h, uint32_t modeNumber);

参数说明：

handle：相机设备的句柄；

modeNumber：读出模式的编号,其取值范围根据读出模式数量决定；

函数说明：

设置相机的读出模式，执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t ret = QHYCCD_ERROR;
ret = SetQHYCCDReadMode(camhandle, 0);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set Read Mode successfully.\n");
}
```

11.uint32_t SetQHYCCDStreamMode(qhyccd_handle *handle, uint8_t mode);

参数说明：

handle：相机设备的句柄；

mode：相机的工作模式，值为 0 时为单帧模式，值为 1 时为连续模式；

函数说明：

设置相机的工作模式，可以设置单帧或者连续模式。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
//设置单帧模式：
int ret = QHYCCD_ERROR;
ret = SetQHYCCDStreamMode(camhandle,0);
if(ret = QHYCCD_SUCCESS)
{
	printf("Set stream mode success!\n" );
}
else
{
	printf("Set stream mode success!\n" );
}
//设置连续模式：
ret = SetQHYCCDStreamMode(camhandle,1);
if(ret = QHYCCD_SUCCESS)
{
	printf("Set stream mode success!\n" );
}
else
{
	printf("Set stream mode success!\n" );
}
```

12.uint32_t InitQHYCCD(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄；

函数说明：

初始化相机参数及 SDK 内部资源，不同相机所需初始化时间不同，执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
int ret = QHYCCD_ERROR;
ret = InitQHYCCD(camhandle);
if(ret == QHYCCD_SUCCESS)
{
	printf("Init QHYCCD success!\n");
}
else
{
	printf("Init QHYCCD fail!\n");
}
```

13.uint32_t GetQHYCCDFWVersion(qhyccd_handle *handle, uint8_t *buf);

参数说明：

handle：相机设备的句柄；

buf：存储固件版本号的变量；

函数说明：

够获取相机驱动(固件)的版本,版本是一个日期,如 18-3-30,若函数成功执行,则返回 QHYCCD_SUCCESS。

目前 QHYCCD 相机分为 WINUSB 和 CYUSB 相机两种，这两种相机的固件版本计算方式有些差异，具体请查看下面的示例代码。

示例代码：

```
int ret = QHYCCD_ERROR;
unsigned char fwv[32];
ret = GetQHYCCDFWVersion(camhandle,fwv);
if(ret == QHYCCD_SUCCESS)
{
	if((fwv[0] >> 4) <= 9)
	{
		printf('Version:20%d-%d-%d', ( (fwv[0]>>4)+0x10,fwv[0]&~0xf0,fwv[1]); //WINUSB cameras
	}
	else
	{
		printf('Version:20%d-%d-%d',fwv[0]>>4,fwv[0]&~0xf0,fwv[1]); //CYUSB cameras
```

```
    }
}
```

14.uint32_t GetQHYCCDSDKVersion(uint32_t *year, uint32_t *month, uint32_t *day, uint32_t *subday);

参数说明：

year：年份；

month：月份；

day：日期；

subday：子版本号；

函数说明：

获取 SDK 的版本号，也就是 qhyccd.dll 的版本，subday 为子版本号，用来区分同一天内的多个版本，可以根据获取到的 SDK 版本判断当前使用的 SDK 是否是最新版本。若函数成功执行，则返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t year,month,day,subday;
ret = GetQHYCCDSDKVersion(&year, &nonth, &day, &subday);
if(ret == QHYCCD_SUCCESS)
{
    printf("%d-%d-%-,%d\n",year,month,day,subday);
    }
else
{
    printf("Get QHYCCD SDK version fail.\n");
}
```

15.uint32_t GetQHYCCDChipInfo(qhyccd_handle *h, double *chipw, double *chiph, uint32_t *imagew, uint32_t *imageh, double *pixelw, double *pixelh, uint32_t *bpp);

参数说明：

handle：相机的设备句柄；

chipw：芯片宽度，单位是毫米；

chiph：芯片高度，单位是毫米；

imagew：图像宽度，单位像素点；

imageh：图像高度，单位像素点；

pixelw：像素宽度，单位是微米；

pixelh：像素高度，单位是微米；

bpp：图像数据位深；

函数说明：

获取相机的芯片信息，包括芯片的尺寸、图像的最大分辨率、像素的尺寸和图像数据的位深。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int ret = QHYCCD_ERROR;
int w,h,bpp;
double chipw,chiph,pixelw,pixelh;
ret = GetQHYCCDChipInfo(camhandle,&chipw,&chiph,&w,&h,&pixelw,&pixelh,&bpp);
if(ret == QHYCCD_SUCCESS)
{
printf("GetQHYCCDChipInfo success!\n");
	printf("CCD/CMOS chip information:\n");
	printf("Chip width : %3f mm\n",chipw);
	printf("Chip height : %3f mm\n",chiph);
	printf("Chip pixel width : %3f um\n",pixelw);
	printf("Chip pixel height : %3f um\n",pixelh);
	printf("image width : %d\n",w);
	printf("image height : %d\n",h);
	printf("Camera depth : %d\n",bpp);
}
else
{
	printf("GetQHYCCDChipInfo failed!\n");
}
```

16.uint32_t GetQHYCCDEffectiveArea(qhyccd_handle *handle, uint32_t *startX, uint32_t *startY, uint32_t *sizeX, uint32_t *sizeY);

参数说明：

handle：相机的设备句柄；

startX：有效区域的起始位置的 x 坐标；

startY：有效区域的起始位置的 y 坐标；

sizeX：有效区域的宽度；

sizeY：有效区域的高度；

函数说明：

这个函数将输出图像有效的尺寸和起始位置，若函数执行成功，则返回 QHYCCD_SUCCESS。起始位置以图像左上角的第一个像素为坐标(0,0)基准点。另外，因为有些相机的图像没有过扫区，所以有效区域和图像尺寸相同。

示例代码：

```
int startx,starty,sizex,sizey;
int ret = QHYCCD_ERROR;
ret = GetQHYCCDEffectiveArea(camhandle,&startx,&starty,&sizex,&sizey);
if(ret == QHYCCD_SUCCESS)
{
	printf("Get camera effective area success.\n");
}
else
{
	printf("Get camera effective area failed.\n");
}
```

17.uint32_t GetQHYCCDOverScanArea(qhyccd_handle *h, uint32_t *startX, uint32_t *startY, uint32_t *sizeX, uint32_t *sizeY);

参数说明：

handle：相机的设备句柄；

startX：图像过扫区起始位置的 x 坐标；

startY：图像过扫区起始位置的 y 坐标；

sizeX：图像过扫区的宽度；

sizeY：图像过扫区的高度；

函数说明：

这个函数将输出过扫区的尺寸和起始位置，若函数执行成功，则返回 QHYCCD_SUCCESS。起始位置以图像左上角的坐标(0,0)为基准点，若获取的过扫区尺寸为 0，则说明该相机不具有过扫区。另外，过扫区的位

置并不是限制在有效区域的某一侧，也有可能位于有效区域其他位置，不同相机的过扫区位置不同，不过同型号的相机过扫区位置是相同的。

示例代码：

```
int startx,starty,sizex,sizey;
int ret = QHYCCD_ERROR;
ret = GetQHYCCDOverScanArea(camhandle, &startx, &starty, &sizex, &sizey);
if(ret == QHYCCD_SUCCESS)
{
	printf("Get camera overscan area success.\n");
}
else
{
	printf("Get camera overscan area failed.\n");
}
```

下图是有效区和过扫区的说明：

18.uint32_t SetQHYCCDBinMode(qhyccd_handle *handle, uint32_t wbin, uint32_t hbin);

参数说明：

handle：相机设备的句柄；

wbin：水平方向的 BIN；

hbin：垂直方向的 BIN；

函数说明：

用来设置相机的 BIN 模式，如 1X1、2X2 等，可以用 IsQHYCCDControlAvailable();函数获取相机支持的 BIN 模式，水平方向和垂直方向的应设置相同的合并倍数，且调用时需要与 SetQHYCCDResolution();函数配合使用。执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
ret = SetQHYCCDBinMode(camhandle,2,2);
if(ret = QHYCCD_SUCCESS)
{
    printf("Set camera bin mode successfully.\n");
}
else
{
    printf("Set camera bin mode fail.\n");
}
```

19.uint32_t SetQHYCCDResolution(qhyccd_handle *handle, uint32_t x, uint32_t y, uint32_t xsize, uint32_t ysize);

参数说明：

handle：相机的设备句柄；

x：起始位置的 x 坐标；

y：起始位置的 y 坐标；

xsize：设置的图像宽度；

ysize：设置的图像高度；

函数说明：

用来设置相机图像的分辨率和 ROI，基准参考点为图像的左上角。函数执行成功时返回

QHYCCD_SUCCESS。

示例代码：

```
ret = SetQHYCCDResolution(camhandle,0,0,500,500);//起始位置为左上角， ROI 尺寸为 500X500
if(ret == QHYCCD_SUCCESS)
{
    printf("Set camera resolution success.\n");
}
else
```

```
{
    printf("Set camera resolution fail.\n");
}
```

20.uint32_t GetQHYCCDCurrentROI(qhyccd_handle *handle, uint32_t *startX, uint32_t *startY, uint32_t *sizeX, uint32_t *sizeY);

参数说明：

handle：相机的设备句柄；

startX：起始位置的 x 坐标；

startY：起始位置的 y 坐标；

sizeX：设置的图像宽度；

sizeY：设置的图像高度；

函数说明：

用来获取相机图像 ROI 的设置值，基准参考点为图像的左上角。函数执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t startX, startY, sizeX, sizeY;
ret = GetQHYCCDCurrentROI(camhandle, &startX, &startY, &sizeX, &sizeY);
if(ret == QHYCCD_SUCCESS)
{
    printf("Get camera resolution success.\n");
}
else
{
    printf("Get camera resolution fail.\n");
}
```

21.uint32_t SetQHYCCDDebayerOnOff(qhyccd_handle *handle, bool onoff);

参数说明：

handle：相机设备句柄；

onoff：彩色模式开启或关闭，true 为开启，false 为关闭；

函数说明：

用来设置彩色相机的彩色模式的开启和关闭，只对彩色相机有效，在调用此函数之前需要先调用 IsQHYCCDControlAvailable 函数判断相机是否是彩色相机。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
ret = SetQHYCCDDebayerOnOff(camhandle,true);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set camera debayer on success.\n");
}
else
{
	printf("Set camera debayer on fail.\n");
}
```

22.uint32_t IsQHYCCDControlAvailable(qhyccd_handle *handle, CONTROL_ID controlId);

参数说明：

handle：相机设备的句柄；

controlId：表示相机功能的枚举变量；

函数说明：

根据 CONTROL_ID 判断相机是否具有某项功能。若相机具有某项功能则返回 QHYCCD_SUCCESS，否则返回 QHYCCD_ERROR。各个 CONTROL_ID 的具体含义请查看 “功能 3.检查相机支持的功能” 的说明。

示例代码：

```
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
if(ret == QHYCCD_SUCCESS)
{
	printf("This camera can setup gain.\n");
	}
else
{
	printf("This camera can setup gain.\n");
}
```

23.uint32_t GetQHYCCDParamMinMaxStep(qhyccd_handle *handle, CONTROL_ID controlId, double *min, double *max, double *step);

参数说明：

handle：相机设备的句柄；

controlId：表示相机功能的枚举变量；

min：用于存储参数设置范围的最小值；

max：用于存储参数设置范围的最大值；

step：用于存储参数设置的最小步长；

函数说明：

可以通过 CONTROL_ID 获取相机参数设置的取值范围，通过此函数可以获取该范围的最大最小值及最小设置步长，执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
double min,max,step;
ret = GetQHYCCDParamMinMaxStep(camhandle, CONTROL_GAIN, &min, &max, &step);
if(ret == QHYCCD_SUCCESS)
{
	printf("min = %lf max = %lf step = %lf\n",min,max,step);
	}
else
{
	printf("Get param min max step fail\n");
}
```

24.double GetQHYCCDParam(qhyccd_handle *handle, CONTROL_ID controlId);

参数说明：

handle：相机设备的句柄；

controlId：表示相机功能的枚举变量；

函数说明：

会根据 CONTROL_ID 获取相机的功能参数的设置值，如设置的曝光时间、增益、偏置等。成功返回相机参数，失败则返回 QHYCCD_ERROR。另外需要注意的是，其中某些功能不是所有相机都支持的，使用前需要使用 IsQHYCCDControlAvailable 函数检查相机是否支持此函数。

示例代码：

```
ret = GetQHYCCDParam(camhandle,CONTROL_EXPOSURE);
if(ret != QHYCCD_ERROR)
{
	printf("The camera's expose time is %d ms.\n",ret/1000);
}
else
{
	printf("Get the camera's expose time fail.\n");
}
```

下为 GetQHYCCDParam 支持的所有 CONTROL_ID：

CONTROL_ID	说明
CONTROL_WBR	获取红色白平衡参数值
CONTROL_WBG	获取绿色白平衡参数值
CONTROL_WBB	获取蓝色白平衡参数值
CONTROL_EXPOSURE	获取曝光时间参数值
CONTROL_GAIN	获取增益参数值
CONTROL_GAINdB	获取 dB 增益参数值（测试中）
CONTROL_OFFSET	获取偏移参数值
CONTROL_GlobalReset	获取全局复位功能参数值
CONTROL_SPEED	获取速度参数值
CONTROL_USBTRAFFIC	获取 Traffic 参数值
CONTROL_VACUUM_PUMP	获取真空泵设置参数值
CONTROL_SensorChamberCycle_PUMP	获取循环泵设置参数值
CONTROL_TRANSFERBIT	获取图像位数参数值
CONTROL_CURTEMP	获取当前温度
CONTROL_CURPWM	获取当前制冷功率
CONTROL_COOLER	获取目标温度设置值
CONTROL_BRIGHTNESS	获取亮度参数值
CONTROL_CONTRAST	获取对比度参数值
CONTROL_GAMMA	获取 Gamma 参数值
CONTROL_AMPV	获取辉光抑制参数值
CONTROL_VCAM	获取 BroadCast WDM 驱动设置参数值
CONTROL_AUTOWHITEBALANCE	获取自动白平衡设置参数值（测试中）
CONTROL_AUTOEXPOSURE	获取自动曝光设置参数值（测试中）
CONTROL_AUTOEXPmessureValue	获取自动曝光阈值设置参数值
CONTROL_AUTOEXPmessureMethod	获取自动曝光遮罩模式设置参数值
CONTROL_ImageStabilization	获取稳像功能设置参数
CAM_CHIPTEMPERATURESENSOR_INTERFACE	获取片上温度传感器参数值

CAM_VIEW_MODE	获取相机的预览模式设置（未启用）
CAM_GPS	获取 GPS 功能参数设置值
CONTROL_CFWSLOTSNUM	获取滤镜轮孔数
CONTROL_CFWPORT	获取滤镜轮当前位置
CONTROL_DDR	获取 DDR 参数设置值（目前 DDR 功能已不再对外开放）
CAM_LIGHT_PERFORMANCE_MODE	获取高低增益参数设置值
CAM_QHY5II_GUIDE_MODE	获取 5II 相机导星模式设置值
DDR_BUFFER_CAPACITY	获取 DDR 缓冲区当前数据量
DDR_BUFFER_READ_THRESHOLD	获取 DDR 缓冲区读出阈值
OutputDataActualBits	获取原始输出数据实际位数
OutputDataAlignment	获取输出数据对齐格式（未启用）
CAM_HUMIDITY	获取相机湿度传感器参数值
CAM_PRESSURE	获取相机压力传感器参数值
CAM_Sensor_ULVO_Status	获取相机 ULVO 状态
CONTROL_RemoveRBI	获取残影消除功能设置参数值
CONTROL_DPC	获取去除热噪声功能设置参数值
CONTROL_DPC_Value	获取去除热噪声功能阈值设置值

25.uint32_t SetQHYCCDParam(qhyccd_handle *handle, CONTROL_ID controlId, double value);

参数说明：

handle：相机设备的句柄；

controlId：表示相机功能的枚举变量；

value：参数设置值；

函数说明：

根据 CONTROL_ID 设置相机功能参数，若函数执行成功，则返回 QHYCCD_SUCCESS。通过 IsQHYCCDControlAvailable 函数可以检查相机是否支持此功能，通过 GetQHYCCDParamMinMaxStep 函数可以获取参数的设置范围和步长。

示例代码：

```
ret = SetQHYCCDParam(camhandle,CONTROL_EXPOSURE,20*1000);
if(ret == QHYCCD_SUCCESS)
```

```
{
    printf("Set camera's expose time success.\n");
    }
else
{
    printf("Set camera's expose time fail.\n");
}
```

下为 SetQHYCCDParam 所有支持的 CONTROL_ID：

CONTROL_ID	说明
CONTROL_WBR	设置红色白平衡
CONTROL_WBG	设置绿色白平衡
CONTROL_WBB	设置蓝色白平衡
CONTROL_EXPOSURE	设置曝光时间
CONTROL_GAIN	设置增益
CONTROL_GAINdB	设置 dB 增益（测试中）
CONTROL_GlobalReset	设置全局复位功能
CONTROL_OFFSET	设置偏移值
CONTROL_SPEED	设置速度
CONTROL_USBTRAFFIC	设置 Traffic
CONTROL_VACUUM_PUMP	设置真空泵开启或关闭
CONTROL_SensorChamberCycle_PUMP	设置循环泵开启或关闭
CONTROL_TRANSFERBIT	设置位数
CONTROL_ROWNOISERE	设置行降噪以减少水平随机条纹
CONTROL_MANULPWM	设置制冷器功率
CAM_GPS	设置 GPS
CAM_IGNOREOVERSCAN_INTERFACE	设置过扫区校正（不建议使用）
QHYCCD_3A_AUTOBALANCE	设置自动白平衡（暂时关闭）
CONTROL_AUTOWHITEBALANCE	设置自动白平衡（测试中）
QHYCCD_3A_AUTOEXPOSURE	设置自动曝光功能（与 CONTROL_AUTOEXPOSURE 相同）
CONTROL_AUTOEXPOSUE	设置自动曝光功能（测试中）
CONTROL_AUTOEXPmessureValue	设置自动曝光阈值（测试中）
CONTROL_AUTOEXPmessureMethod	设置自动曝光遮罩模式（测试中）
CONTROL_ImageStabilization	设置图像稳像功能（测试中）
QHYCCD_3A_AUTOFOCUS	设置自动调焦
CONTROL_BRIGHTNESS	设置亮度
CONTROL_CONTRAST	设置对比度
CONTROL_GAMMA	设置 Gamma
CONTROL_AMPV	设置辉光抑制开启或关闭

CONTROL_COOLER	设置制冷器目标温度
CONTROL_VCAM	设置 BroadCast WDM 开启或关闭
CAM_VIEW_MODE	设置预览模式（未启用）
CONTROL_CFWPORT	设置滤镜轮目标孔位
CONTROL_DDR	设置 DDR 开启或关闭（目前已关闭 DDR 设置功能）
CAM_LIGHT_PERFORMANCE_MODE	设置高低增益切换
CAM_QHY5II_GUIDE_MODE	设置 5II 系列相机导星模式开启或关闭
CONTROL_ImgProc	设置图像处理的镜像或旋转操作
CONTROL_RemoveRBI	设置残影消除功能
CONTROL_DPC	设置去除热噪声功能（测试中）
CONTROL_DPC_value	设置去除热噪声功能阈值（测试中）

26.uint32_t GetQHYCCDMemLength(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄；

函数说明：

返回相机图像的最大所需内存长度，此函数返回的内存长度为固定值，且会比实际所需值稍大一些以避免一些意外的内存溢出问题，其计算方式为:

黑白相机：(imagew+100)*(imageh+100)*2

彩色相机：(imagew+100)*(imageh+100)*3

示例代码：

```
uint32_t length = 0;
length = GetQHYCCDMemLength(camhandle);
if(length > 0)
{
    printf("Get memory length successfully.\n");
}
else
{
    printf("Get memory length failed.\n");
}
```

27.uint32_t ExpQHYCCDSingleFrame(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄；

函数说明：

开始单帧模式曝光，调用后相机会开始曝光一张单帧图像。若函数执行成功，则返回 QHYCCD_SUCCESS，个别相机可能会返回其他值，正常情况下返回值不是 QHYCCD_ERROR 即可视为执行成功。

示例代码：

```
int ret = QHYCCD_ERROR;
ret = ExpQHYCCDSingleFrame(camhandle);
if(ret = QHYCCD_SUCCESS)
{
	printf(“Camera expose success.\n” );
	}
else
{
	printf(“Camera expose failed.\n” );
}
```

28.uint32_t GetQHYCCDSingleFrame(qhyccd_handle *handle, uint32_t *w, uint32_t *h, uint32_t *bpp, uint32_t *channels, uint8_t *imgdata);

参数说明：

handle：相机的设备句柄；

w：用于存储获取到的图像的宽度信息；

h：用于存储获取到的图像的高度信息；

bpp：用于存储获取到的图像的位数信息；

channels：用于存储获取到的图像的通道数；

imgdata：用于存储获取到的图像数据；

函数说明：

从相机中获取一帧图像数据，获取的数据存储在 ImgData 中。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int ret = QHYCCD_ERROR;
ret = GetQHYCCDSingleFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
if(ret == QHYCCD_SUCCESS)
{
```

```
    printf("Get camera single frame succeess.\n" );
}
else
{
    printf("Get camera single frame failed.\n" );
}
```

29.uint32_t CancelQHYCCDExposingAndReadout(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄；

函数说明：

停止相机曝光并且停止数据读出。停止时要保证软件和相机同步，相机不输出数据且软件不接收数据。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int ret = QHYCCD_ERROR;
ret = CancelQHYCCDExposingAndReadout(camhandle);
if(ret == QHYCCD_SUCCESS)
{
    printf("Cancel camera expose and readout success.\n" );
}
else
{
    printf("Cancel camera expose and readout failed.\n" );
}
```

30.uint32_t CancelQHYCCDExposing(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄；

函数说明：

停止相机曝光。对 CYUSB 的相机来说，这两个停止曝光的函数是相同的，停止曝光的同时也会停止数据读出，但是对 WINUSB 的相机来说这个函数仅停止曝光时间，图像数据仍需要读出。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int ret = QHYCCD_ERROR;
ret = CancelQHYCCDExposing(camhandle);
if(ret == QHYCCD_SUCCESS)
{
    printf("Cancel camera expose success!\n" );
}
else
```

```
{
    printf("Cancel camera expose failed.\n" );
}
```

31.uint32_t BeginQHYCCDLive(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄；

函数说明：

开始连续模式曝光，曝光开始后会持续输出视频流数据，上位机需要持续调用 GetQHYCCDLiveFrame 函数以读出图像数据。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int ret = QHYCCD_ERROR;
ret = BeginQHYCCDLive(camhandle);
if(ret = QHYCCD_SUCCESS)
{
    printf("Camera begin live success.\n" );
}
else
{
    printf("Camera begin live failed.\n" );
}
```

32.uint32_t GetQHYCCDLiveFrame(qhyccd_handle *handle, uint32_t *w, uint32_t *h, uint32_t *bpp, uint32_t *channels, uint8_t *imgdata);

参数说明：

handle：相机的设备句柄；

w：用于存储获取到的图像的宽度信息；

h：用于存储获取到的图像的高度信息；

bpp：用于存储获取到的图像的位数信息；

channels：用于存储获取到的图像的通道数；

imgdata：用于存储获取到的图像数据；

函数说明：

从相机中获取图像数据，获取的数据存储在 ImgData 中。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int ret = QHYCCD_ERROR;
ret = GetQHYCCDLiveFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
if( == QHYCCD_SUCCESS)
{
	printf(“ Get camera live frame succeess.\n” );
}
else
{
	printf(“ Get camera live frame failed.\n” );
}
```

33.uint32_t StopQHYCCDLive(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄；

函数说明：

停止相机的连续模式。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
ret = StopQHYCCDLive(camhandle);
if(ret == QHYCCD_SUCCESS)
{
	printf("Stop camera live success.\n");
}
else
{
	printf("Stop camera live fail.\n");
}
```

34.uint32_t ControlQHYCCDTemp(qhyccd_handle *handle, double targettemp);

参数说明：

handle：相机设备的句柄；

targettemp：相机目标温度；

函数说明：

设置相机制冷的目标温度，和 SetQHYCCDParam 的 CONRTOL_COOLER 功能相同，成功执行则返回 QHYCCD_SUCCESS。使用前可调用 IsQHYCCDControlAvailable();函数判断相机是否具有制冷功能。

示例代码：

```
double temp = 0;
ret = ControlQHYCCDTemp(camhandle,temp);
if(ret == QHYCCD_SUCCESS)
{
	printf("Control camera temperature success.\n");
}
else
{
	printf("Control camera temperature fail.\n");
}
```

35.uint32_T IsQHYCCDCFWPlugged(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄；

函数说明：

检查滤镜轮是否已连接，只有 QHY5IIICOOL 系列和 MINICAM5F_M 实现了此函数，若返回 QHYCCD_SUCCESS，则视为已连接。

示例代码：

```
ret = IsQHYCCDCFWPlugged(camhandle);
if(ret == QHYCCD_SUCCESS)
{
	printf(“ CFW has been connected.\n” );
}
else
{
	printf(“ CFW didn’ t be connected.\n” );
}
```

36.uint32_t SendOrder2QHYCCDCFW(qhyccd_handle *handle, char *order, uint32_t length);

参数说明：

handle：相机返回的设备句柄；

order：滤镜轮的目标位置；

length：order 的字符长度，通常为 1；

函数说明：

控制滤镜轮转动到指定位置，函数执行成功时返回 QHYCCD_SUCCESS；

示例代码：

```
char order = ‘ 0’ ;
ret = SendOrder2QHYCCDCFW(camhandle,&order,1);
if(ret = QHYCCD_SUCCESS)
{
      printf(“ Set CFW success.\n” );
}
else
{
      printf(“ Set CFW error.\n” );
}
```

37.uint32_t GetQHYCCDCFWStatus(qhyccd_handle *handle, char *status);

参数说明：

handle：相机的设备句柄；

status：滤镜轮的状态信息；

函数说明：

获取滤镜轮当前的位置信息，既当前孔位，函数执行成功则返回 QHYCCD_SUCCESS。此函数和 GetQHYCCDParam 函数的 CONTROL_CFWPORT 功能相同，但返回值会有所差异，此函数获取到的结果是 char 类型的数据，而 GetQHYCCDParam 获取到的是 double 类型的数据，若在 C#环境下进行二次开发使用此函数会无法正确获取数据，因此使用 C#进行二次开发时建议使用 GetQHYCCDParam 函数。

示例代码：

```
char status;
ret = GetQHYCCDCFWStatus(camhandle, &status);
if(ret = QHYCCD_SUCCESS)
{
      printf(“Now position is %c.\n”, status );
}
else
```

```
{
    printf(“ Get QHYCCD CFW status error.\n” );
}
```

38.uint32_t SetQHYCCDGPSVCOXFreq(qhyccd_handle *handle, uint16_t i);

参数说明：

handle：相机的设备句柄；

i：VCOX 的频率；

函数说明：

用来控制 GPS 相机的 VCOX 频率，暂时只有 QHY174-GPS 支持此函数。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int i = 100;
ret = SetQHYCCDGPSVCOXFreq(camhandle,i);
if(ret == QHYCCD_SUCCESS)
{
    printf("Set QHYCCD VCOX frequency success.\n");
}
else
{
    printf("Set QHYCCD VCOX frequency fail.\n");
}
```

39.uint32_t SetQHYCCDGPSLedCalMode(qhyccd_handle *handle, uint8_t i);

参数说明：

handle：相机的设备句柄；

i：LED 灯的开关和模式选择，0 为关闭，1 为从模式，2 为主模式；

函数说明：

设置 LED 灯校准的开关与校准模式，参数为 0 时关闭 LED，参数为 1 时为从模式校准，参数为 2 时为主模式校准，一般只在关闭 LED 时需要调用此函数，设置 Position A 和 Position B 时会自动选择校准模式，暂时只有 QHY174-GPS 支持此函数。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int i = 2;
ret = SetQHYCCDGPSLedCalMode(camhandle,i);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set QHYCCD led cal mode success.\n");
}
else
{
	printf("Set QHYCCD led cal mode fail.\n");
}
```

40.void SetQHYCCDGPSPOSA(qhyccd_handle *handle, uint8_t is_slave, uint32_t pos, uint8_t width);

参数说明：

handle：相机的设备句柄；

is_slave：相机的工作模式，0 为主模式，1 为从模式；

pos：脉冲位置；

width：脉冲宽度，通常固定为 54；

函数说明：

设置 LED 脉冲位置，用于快门曝光。当改变曝光时间时，需要设置这个位置。测量电路将使用这个位置作为快门启动时间，暂时只有 QHY174-GPS 支持此函数。

示例代码：

```
int pos = 1000,width = 54;
SetQHYCCDGPSPOSA(camhandle,pos,width);
```

41.void SetQHYCCDGPSPOSB(qhyccd_handle *handle, uint8_t is_slave, uint32_t pos, uint8_t width);

参数说明：

handle：OpenQHYCCD();返回的相机句柄；

is_slave：取决于相机使用的是那种模式，0：主模式，1：从模式；

pos：设置 LED 脉冲位置；

width：LED 脉冲宽度，通常设置为 54；

函数说明：

设置 LED 脉冲位置，用于快门曝光。当改变曝光时间时，需要设置这个位置。测量电路将使用这个位置作为快门结束时间，暂时只有 QHY174-GPS 支持此函数。

示例代码：

```
int pos = 10000,width = 54;
SetQHYCCDGPSPOSA(camhandle,pos,width);
```

42.uint32_t SetQHYCCDGPSMasterSlave(qhyccd_handle *handle, uint8_t i);

参数说明：

handle：相机的设备句柄；

i：设置相机的主从模式，0 为主模式，1 为从模式；

函数说明：

用来控制 GPS 相机的主从模式，暂时只有 QHY174-GPS 支持此函数。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int i = 0;
ret = SetQHYCCDGPSMasterSlave(camhandle,i);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set QHYCCD GPS master slave success.\n");
}
else
{
	printf("Set QHYCCD GPS master slave fail.\n");
}
```

43.void SetQHYCCDGPSSlaveModeParameter(qhyccd_handle *handle, uint32_t target_sec, uint32_t target_us, uint32_t deltaT_sec, uint32_t deltaT_us, uint32_t expTime);

参数说明：

handle：相机的设备句柄；

target_sec：开始时间，单位为 s，QHYCCD 定义的“JS”，它指的是一段时间；

target_us：开始时间，单位为 us；

deltaT_sec：间隔，单位为 s；

deltaT_us：间隔，单位为 us；

expTime：曝光时间，单位是 us；

函数说明：

设置从模式下的拍摄参数，暂时只有 QHY174-GPS 支持此函数。若执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

设定时间为 2017.10.19 0:30， 我们可以获得 JS 为 695925030， 现在想让相机在十分钟后开始曝光（ 600s），曝光时间为 100ms，曝光间隔为 200ms。

```
target_sec=695925030+600;
target_us=0;
deltaT_sec=0
deltaT_us=200*1000;
expTime=100*1000;
SetQHYCCDGPSSlaveModeParameter(camhandle, target_sec, target_us, deltaT_sec, deltaT_us, expTime) ;
```

44.uint32_t SetQHYCCDEnableLiveModeAntiRBI(qhyccd_handle *h, uint32_t value);

参数说明：

h：相机的设备句柄；

value：开启或关闭 AntiRBI 模式，0x1C00；

函数说明：

用来开启相机的 AntiRBI 模式，开启后图像会以一亮一暗的形式向外输出图像，此模式可以消除图像残影的影响，函数执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
ret = SetQHYCCDEnableLiveModeAntiABI(camhandle, 0x1C00);
if (ret == QHYCCD_SUCCESS)
{
	printf("Enable Live Mode AntiRBI successfully.\n");
}
else
{
	printf("Enable Live Mode AntiRBI failed.\n");
}
```

45.uint32_t EnableQHYCCDBurstMode(qhyccd_handle *h,bool i);

参数说明：

h：相机的设备句柄；

i：开启或关闭 Burst 模式，true 为开启，false 为关闭；

函数说明：

设置 Burst 模式开启或关闭，执行成功时返回 QHYCCD_SUCCESS。Burst 模式是连续模式的子模式，在连续模式下才能设置此模式，当使能 Burst 模式时，相机会暂停输出视频流，并等待开始命令，当接收到开始命令之后，相机会根据设置输出预先设定的图像。

示例代码：

```
ret = EnableQHYCCDBurstMode(camhandle, true);
if(ret == QHYCCD_SUCCESS)
{
	printf("Enable Burst Mode Successfully.\n");
}
else
{
	printf("Enable Burst Mode Failed.\n");
}
```

46.uint32_t EnableQHYCCDBurstCountFun(qhyccd_handle *h,bool i);

参数说明：

h：相机的设备句柄；

i：开启或关闭 Burst 模式计数功能，true 为开启，false 为关闭；

函数说明：

开启或关闭 Burst 模式下的计数功能，执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
ret = EnableQHYCCDBurstCountFun(camhandle, true);
if(ret == QHYCCD_SUCCESS)
{
	printf("Enable Burst Mode Count Function successfully.\n");
}
else
{
	printf("Enable Burst Mode Count Function failed.\n");
}
```

47.uint32_t ResetQHYCCDFrameCounter(qhyccd_handle *h);

参数说明：

h：相机的设备句柄；

函数说明：

重置计数器的值为 0，Burst 模式下每次调用 ReleaseQHYCCDBurstIDLE 函数时都会将计数器清零，因此 Burst 模式下不必执行此函数，执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
ret = ResetQHYCCDFrameCounter(camhandle);
if(ret == QHYCCD_SUCCESS)
{
	printf("Reset Frame Counter Successfully.\n");
}
else
{
	printf("Reset Frame Counter Failed.\n");
}
```

48.uint32_t SetQHYCCDBurstModeStartEnd(qhyccd_handle *h, unsigned short start, unsigned short end);

参数说明：

h：相机的设备句柄；

start：图像输出的开始帧；

end：图像输出的结束帧；

函数说明：

会根据设置输出指定的图像，比如设置 start 为 1，end 为 4，会输出第 2 和第 3 帧图像，执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
ret = SetQHYCCDBurstModeStartEnd(camhandle, 1, 4);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set Burst Mode Start and End Successfully.\n");
}
else
{
	printf("Set Burst Mode Start and End Failed.\n");
}
```

49.uint32_t SetQHYCCDBurstIDLE(qhyccd_handle *h);

参数说明：

h：相机的设备句柄；

函数说明：

设置相机进入 IDLE 状态，执行成功时返回 QHYCCD_SUCCESS。此函数需要和 ReleaseQHYCCDBurstIDLE 配合使用。

示例代码：

```
ret = SetQHYCCDBurstIDLE(camhandle);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set Burst Mode IDLE Successfully.\n");
}
else
{
	printf("Set Burst Mode IDLE Failed.\n");
}
```

50.uint32_t ReleaseQHYCCDBurstIDLE(qhyccd_handle *h);

参数说明：

h：相机的设备句柄；

函数说明：

需要和 SetQHYCCDBurstIDLE 函数配合使用，可以解除相机 Burst 模式下的 IDLE 状态，解除 IDLE 状态后，相机会输出预先设定的图像。

示例代码：

```
ret = ReleaseQHYCCDBurstIDLE(camhandle);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set Burst Mode IDLE Successfully.\n");
}
else
{
	printf("Set Burst Mode IDLE Failed.\n");
}
```

51.uint32_t SetQHYCCDBurstModePatchNumber(qhyccd_handle *h,uint32_t value);

参数说明：

h：相机的设备句柄；

value：数据包的大小；

函数说明：

用来在 Burst 模式下补充数据包，避免数据不够无法输出图像。

示例代码：

```
ret = SetQHYCCDBurstModePatchNumber(camhandle, 32001);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set Burst Mode patch number Successfully.\n");
}
else
{
	printf("Set Burst Mode patch number Failed.\n");
}
```

52.uint32_t GetQHYCCDTrigerInterfaceNumber(qhyccd_handle *handle, uint32_t *modeNumber)

参数说明：

handle：相机的设备句柄；

modeNumber：接口的数量；

函数说明：

获取相机触发接口的数量，有些相机支持多种触发接口，通过此函数可以知道接口的数量。

示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
uint32_t number;
retVal = GetQHYCCDTrigerInterfaceNumber(camhandle, &number);
if(ret == QHYCCD_SUCCESS)
{
    printf("Get interface number successfully.\n");
}
```

53.uint32_t GetQHYCCDTrigerInterfaceName(qhyccd_handle *handle, uint32_t modeNumber, char *name)

参数说明：

handle：相机的设备句柄

modeNumber：接口名称的下标

name：接口的名称

函数说明：

获取相机所支持的触发接口的名称。

示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
char name[40] = { 0 };
for(int i = 0; i < number; i++)
{
    retVal = GetQHYCCDTrigerInterfaceName(camhandle, i, name);
```

```
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get triger interface name successfully.\n");
    }
}
```

54.uint32_t SetQHYCCDTrigerInterface(qhyccd_handle *handle, uint32_t trigerMode)

参数说明：

handle：相机的设备句柄

trigerMode：相机的触发接口

函数说明：

设置相机的触发接口，当相机支持多种触发接口时可以通过此函数进行设置。

示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
retVal = SetQHYCCDTrigerInterface(camhandle, 0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set triger mode successfully.\n");
}
```

55.uint32_t SetQHYCCDTrigerMode(qhyccd_handle *handle, uint32_t trigerMode);

参数说明：

handle：相机的设备句柄

trigerMode：相机的触发模式

函数说明：

设置相机的触发功能模式，当参数为 0 时，为关闭外触发功能，与 SetQHYCCDTrigerFunction(camhandle, false)功能相同。当相机支持多种触发模式时，可以通过此函数设置触发功能的工作模式。目前功能未完善，可以暂时忽略。

示例代码：

```
ret = SetQHYCCDTrigerMode(camhandle, 0);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set Trigger Mode Successfully.\n");
}
```

56.uint32_t SetQHYCCDTrigerFunction(qhyccd_handle, bool value);

参数说明：

handle：相机的设备句柄；

value ：开启或关闭外触发功能，true 为开启，false 为关闭；

函数说明：

设置相机的外触发功能开启或关闭，外触发开启时，相机会进入 Trigger In 模式，此模式下相机不会立即开始曝光，而是会等待外部触发信号。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
ret = SetQHYCCDTrigerFunction(camhandle,true);
if(ret == QHYCCD_SUCCESS)
{
	printf(" Open QHYCCD triger success.\n" );
}
else
{
	printf(" Open QHYCCD triger fail.\n" );
}
```

57.uint32_t EnableQHYCCDTrigerOut(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄

函数说明：

使能相机的 Trigger Out 功能，使能时，相机会输出曝光相关的测量波形。注意，QHY4040 和 QHY4040PRO 由于硬件原因，无法同时使用 Trigger In 和 Trigger Out 功能。

示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
retVal = EnableQHYCCDTrigerOut(camhandle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Enable trigger out successfully.\n");
}
```

58.uint32_t EnableQHYCCDTrigerOutA(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄

函数说明：

使能相机的 Trigger Out 功能，并以模式 A 输出触发信号，使能时，相机会输出曝光相关的测量波形。

示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
retVal = EnableQHYCCDTrigerOutA(camhandle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Enable trigger out successfully.\n");
}
```

59.uint32_t SendSoftTriger2QHYCCDCam(qhyccd_handle *handle)

参数说明：

handle：相机的设备句柄

函数说明：

当相机处于 Trigger In 模式时，可以通过此函数向相机发送软件触发信号，触发相机拍摄。

示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
retVal = SendQHYCCDTriger2QHYCCDCam(camhandle);
```

```
if(retVal == QHYCCD_SUCCESS)
{
    printf("Send software trigger signal successfully.\n");
}
```

60.uint32_t SetQHYCCDTrigerFilterOnOff(qhyccd_handle *handle, bool onoff)

参数说明：

handle：相机的设备句柄

onoff：开启或关闭滤波

函数说明：

设置开启或关闭滤波功能，通过滤波功能可以屏蔽机械开关造成的电源波动影响，默认情况下是开启的，滤波时间为 100ms。

示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
retVal = SetQHYCCDTrigerFilterOnOff(camhandle, false);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Turn trigger filter off successfully.\n");
}
```

61.uint32_t SetQHYCCDTrigerFilterTime(qhyccd_handle *handle, uint32_t time)

参数说明：

handle：相机的设备句柄

time：滤波时间，单位为 ms，设置范围为 1~100000ms

函数说明：

设置触发模式的滤波时间，默认为 100ms

示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
retVal = SetQHYCCDTrigerFilterTime(camhandle, 150);
```

```
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set trigger filter time successfully.\n");
}
```

62. uint32_t EnableQHYCCDImageOSD(qhyccd_handle *h,uint32_t i);

参数说明：

h：相机的设备句柄；

i：是否显示文本，0 为不显示，1 为显示帧序号，2 为显示 GPS 信息；

函数说明：

可以将文本显示在图像的左上角，如帧序号、GPS 信息等，执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
ret = EnableQHYCCDImageOSD(camhandle, 1);
if(ret == QHYCCD_SUCCESS)
{
    printf("Enable display image sequence number successfully.\n");
}
else
{
    printf("Enable display image sequence number failed.\n");
}
```

63.void EnableQHYCCDMessage(bool enable);

参数说明：

enable：开启或关闭 debug 输出；

函数说明：

开启或关闭 SDK 内部的 debug 信息输出。设置信息输出时可以通过编译器或 DebugView 捕获，DebugView 捕获关键词为 QHYCCD。

示例代码：

```
EnableQHYCCDMessage(true);
```

64.void RegisterPnpEventIn(void (*in_pnp_event_in_func)(char *id))

参数说明：

in_pnp_event_in_func：注册函数的地址指针

函数说明：

注册相机接入事件，当相机连接的时候 SDK 会调用上位机中注册的函数。

示例代码：

```
void pnp_Event_In_Func(char *id)
{
    printf("Camera In.\n");
}

RegisterPnpEventIn(pnp_Event_In_Func);
```

65.void RegisterPnpEventOut(void (*in_pnp_event_out_func)(char *id))

参数说明：

in_pnp_event_out_func：注册函数的地址指针

函数说明：

注册相机断开时间，当相机断开的时候 SDK 会调用上位机中注册的函数。

示例代码：

```
void pnp_Event_Out_Func(char *id)
{
    printf("Camera Out.\n");
}

RegisterPnpEventOut(pnp_Event_Out_Func);
```

66.uint32_t SetQHYCCDTwoChannelCombineParameter(qhyccd_handle *handle, double x,double ah,double bh,double al,double bl);

参数说明：

handle：相机的设备句柄

x：高低增益切换点

ah：高增益通道斜率系数

bh：高增益通道截距系数

al：低增益通道斜率系数

bl：低增益通道结局系数

函数说明：

适用于有高低增益通道的相机，当使用高低增益通道拼接的方式输出 16 位数据时，可能会因为温度影响而影响效果，无法达到高低增益平滑过渡的需求，此时可以使用此函数进行调节以保证拼接的准确性和线性度。函数执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t ret = QHYCCD_ERROR;
ret = SetQHYCCDTwoChannelCombineParameter(camhandle, 4096, 0, 30000, 16, 30000);
if(ret == QHYCCD_SUCCESS)
{
	printf(“Setup combine parameter successfully.\n”);
}
else
{
	printf(“Setup combine parameter failed.\n”);
}
```

67.uint32_t GetQHYCCDPreciseExposureInfo(qhyccd_handle *h, uint32_t *PixelPeriod_ps, uint32_t *LinePeriod_ns, uint32_t *FramePeriod_us, uint32_t *ClocksPerLine, uint32_t *LinesPerFrame, uint32_t *ActualExposureTime, uint8_t *isLongExposureMode);

参数说明：

h：相机的设备句柄

PixelPeriod_ps：像素周期时间长度，单位为皮秒

LinePeriod_ns：行周期时间长度，单位为纳秒

FramePeriod_us：帧周期时间长度，单位为微秒，单帧连续模式以及不同读出模式下会有差异

ClocksPerLine：每行的时钟数

LinesPerFrame：每帧图像的行数

ActualExposureTime：实际曝光时间，单位为微秒

isLongExposureMode：长短曝光标志位，0 为短曝光，1 为长曝光，判断依据为是否大于帧周期

函数说明：

获取精确的曝光时间相关信息，包括行周期、帧周期、时机曝光时间等。函数执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t PixelPeriod_ps, LinePeriod_ns, FramePeriod_us, ClocksPerLine, LinesPerFrame, ActualExposureTime;
uint8_t isLongExposureMode;

ret = GetQHYCCDPreciseExposureInfo(camhandle, &PixelPeriod_ps, &LinePeriod_ns, &FramePeriod_us,
&ClocksPerLine, &LinesPerFrame, &ActualExposureTime, &isLongExposureMode);
if(ret == QHYCCD_SUCCESS)
{
    printf("Get precise exposure information successfully.\n");
}
else
{
    printf("Get precise exposure information failed.\n");
}
```

68.uint32_t GetQHYCCDRollingShutterEndOffset(qhyccd_handle *h,uint32_t row,double *offset);

参数说明：

h：相机的设备句柄

row：需要计算的图像行数，取值范围为 0 到图像行数减 1

offset：返回的偏移值计算结果

函数说明：

根据图像行数计算曝光时间的偏移值，此函数的返回值和 GPS 时间配合使用可以得到图像每行曝光的精确时间。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
double offset;
ret = GetQHYCCDRollingShutterEndOffset(camhandle, 1000, &offset);
if(ret == QHYCCD_SUCCESS)
{
    printf("Enable display image sequence number successfully.\n");
}
else
{
    printf("Enable display image sequence number failed.\n");
}
```

69.void QHYCCDSensorPhaseReTrain(qhyccd_handle *handle);

参数说明：

handle：相机的设备句柄

函数说明：

解决相机因为相位导致的图像条纹问题，调用此函数相机内部会重新计算相位值。

示例代码：

```
QHYCCDSensorPhaseReTrain(camhandle);
```

70.uint32_t GetQHYCCDImageStabilizationGravity(qhyccd_handle *handle, int *GravityX, int *GravityY);

参数说明：

handle：相机的设备句柄

GravityX：图像重心的 X 坐标

GravityY：图像重心的 Y 坐标

函数说明：

启用稳像功能后，可以使用此函数获取图像重心的坐标位置。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
int GravityX = 0, GravityY = 0;
uint32_t ret = QHYCCD_ERROR;
ret = GetQHYCCDImageStabilizationGravity(camhandle, &GravityX, &GravityY);
```

```
if(ret == QHYCCD_SUCCESS)
    printf(”Get Image Gravity success\n”);
else
    printf(”Get Image Gravity failed\n”);
```

71.uint32_t GetQHYCCDSensorName(qhyccd_handle *handle, char *name);

参数说明：

handle：相机的设备句柄

name：传感器的名称

函数说明：

获取传感器芯片的名称，如 IMX411。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
char name[20] = { 0 };
uint32_t ret = QHYCCD_ERROR;
ret = GetQHYCCDSensorName(camhandle, name);
if(ret == QHYCCD_SUCCESS)
    printf(“Get Sensor Name Success %s\n”, name);
else
    printf(“Get Sensor Name Failed\n”);
```

72.uint32_t QHYCCD_DbGainToGainValue(qhyccd_handle *h, double dbgain, double *gainvalue);

参数说明：

handle：相机的设备句柄

dbgain：dB 增益值

gainvalue：增益设置值

函数说明：

根据 dB 增益值换算成 SDK 中的增益设置值。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
double gainvalue;
```

```
uint32_t ret = QHYCCD_ERROR;
ret = QHYCCD_DbGainToGainValue(camhandle, 100, &gainvalue);
if(ret == QHYCCD_SUCCESS)
printf("call function success\n");
else
printf("call function failed\n");
```

73.uint32_t QHYCCD_GainValueToDbGain(qhyccd_handle *h, double gainvalue, double *dbgain);

参数说明：

handle：相机的设备句柄

gainvalue：增益设置值

dbgain：dB 增益值

函数说明：

根据 SDK 中的增益设置值换算成 dB 增益值。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
double dbgaine;
uint32_t ret = QHYCCD_ERROR;
ret = QHYCCD_GainValueToDbGain(camhandle, 100, &dbgain);
if(ret == QHYCCD_SUCCESS)
printf("call function success\n");
else
printf("call function failed\n");
```

74.uint32_t QHYCCD_curveFullWell(qhyccd_handle *handle, double gainV, double *fullwell);

参数说明：

handle：相机的设备句柄

gainV：增益值

fullwell：满井值

函数说明：

根据增益设置值获取满井曲线对应值。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
double fullwell;
uint32_t ret = QHYCCD_ERROR;
ret = QHYCCD_curveFullWell(camhandle, 100, &fullwell);
if(ret == QHYCCD_SUCCESS)
printf("call function success\n");
else
printf("call function failed\n");
```

75.uint32_t QHYCCD_curveReadoutNoise(qhyccd_handle *handle,double gainV,double *readoutnoise)

参数说明：

handle：相机的设备句柄

gainV：增益值

readoutnoise：读出噪声

函数说明：

根据增益值获取读出噪声曲线对应的值。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
double readoutnoise;
uint32_t ret = QHYCCD_ERROR;
ret = QHYCCD_curveReadoutNoise(camhandle, 100, &readoutnoise);
if(ret == QHYCCD_SUCCESS)
printf("call function success\n");
else
printf("call function failed\n");
```

76.uint32_t QHYCCD_curveSystemGain(qhyccd_handle *handle,double gainV,double *systemgain)

参数说明：

handle：相机的设备句柄

gainV：增益值

systemgain：系统增益值

函数说明：

根据增益值获取系统增益曲线对应的值。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
double systemgain;
uint32_t ret = QHYCCD_ERROR;
ret = QHYCCD_curveSystemGain(camhandle, 100, &systemgain);
if(ret == QHYCCD_SUCCESS)
printf("call function success\n");
else
printf("call function failed\n");
```

77.uint32_t SetQHYCCDFrameDetectOnOff(qhyccd_handle *handle, bool onoff)

参数说明：

handle：相机的设备句柄

onoff：开启或关闭帧监测功能

函数说明：

设置相机开启帧监测功能，可以通过此功能监测数据是否在传输过程中发生意外，其工作原理为在图像数据指定位置设置一个指定值以替换部分数据，SDK 获取到图像数据之后会对指定位置的数据进行检测，如数据不符合设置值则判定为图像获取失败并将此图像过滤掉。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t ret = QHYCCD_ERROR;
ret = SetQHYCCDFrameDetectOnOff(camhandle, true);
if(ret == QHYCCD_SUCCESS)
printf("call function success\n");
else
printf("call function failed\n");
```

78.uint32_t SetQHYCCDFrameDetectCode(qhyccd_handle *handle, uint8_t code)

参数说明：

handle：相机的设备句柄

code：数据校验值

函数说明：

设置帧监测功能数据校验值。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t ret = QHYCCD_ERROR;
ret = SetQHYCCDFrameDetectCode(camhandle, 0);
if(ret == QHYCCD_SUCCESS)
printf(“call function success\n”);
else
printf(“call function failed\n”);
```

79.uint32_t SetQHYCCDFrameDetectPos(qhyccd_handle *handle, uint32_t pos)

参数说明：

handle：相机的设备句柄

pos：数据位置

函数说明：

设置帧监测功能数据校验位置。执行成功时返回 QHYCCD_SUCCESS。

示例代码：

```
uint32_t ret = QHYCCD_ERROR;
ret = SetQHYCCDFrameDetectPos(camhandle, 0);
if(ret == QHYCCD_SUCCESS)
printf(“call function success\n”);
else
printf(“call function failed\n”);
```

三、功能设置介绍

相机控制的标准流程为：1.连接相机→2.控制相机→3.断开相机。连接操作用于打开相机并进行相关初始设置，控制操作用于控制相机的各项功能，包括参数设置，制冷、CFW、GPS、Trigger、Burst 等功能控制等，断开相机用来关闭相机并释放相关资源。

1.获取相机句柄

扫描设备上连接的相机的数量和 ID，并根据 ID 打开相机，获得设备句柄。

用到的函数有 InitQHYCCDResource、ScanQHYCCD、GetQHYCCDId、OpenQHYCCD，下为获取相机句柄的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
char camId[40] = {0};
qhyccd_handle *camHandle = NULL;

//初始化 SDK 内部的资源
retVal = InitQHYCCDResource();
if(retVal == QHYCCD_SUCCESS)
{
    printf("Initialize SDK resource successfully.\n");
}
else
{
    printf("Initialize SDK resource failed.\n");
    return -1;
}

//扫描相机，并返回设备数量
camNum = ScanQHYCCD();
if(camNum == 0)
{
    printf("Didn't find QHYCCD cameras.\n");
    return -1;
}

//连接 QHY 相机列表里的第一个设备
for(int i = 0; i < camNum; i ++)
{
    retVal = GetQHYCCDId(i, camId); //获取设备 ID
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get camera's ID successfully.\n");
        break;
```

```
    }
    else
    {
        printf("Get camera's ID failed.\n");
        return -1;
    }
}

camHandle = OpenQHYCCD(camId); //根据设备 ID 打开设备，并返回设备句柄
if(camHandle != NULL)
{
    printf("Openc camera successfully.\n");
}
else
{
    printf("Openc camera failed.\n");
    return -1;
}
```

2.设置读出模式

设置相机的读出模式，不同型号的相机有不同的读出模式，相机在不同读出模式下会有不同的特性。

获取相机句柄之后初始化之前可以进行此设置，使用的函数有 GetQHYCCDNumberOfReadMode、GetQHYCCDReadModeName、GetQHYCCDReadModeResolution、SetQHYCCDReadMode，下为设置读出模式的示例代码：

```
//获取相机读出模式的数量，返回 1 则说明只有一个标准读出模式（STANDARD MODE）
retVal = GetQHYCCDNumberOfReadMode(camHandle,readModeNum);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Get Read Mode number successfully.\n");
}
else
{
    printf("Get Read Mode number failed.\n");
    return -1;
}

for(int i = 0; i < readModeNum; i++)
{
    //获取读出模式的名称
    retVal = GetQHYCCDReadModeName(camHandle, i, readModeName);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get Read Mode name successfully.\n");
    }
    else
```

```
    {
        printf("Get Read Mode name failed.\n");
        return -1;
    }

    //获取当前读出模式的分辨率
    retVal = GetQHYCCDReadModeResolution(camHandle, i, readModeWidth,readModeHeight);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get Read Mode resolution successfully.\n");
    }
    else
    {
        printf("Get Read Mode resolution failed.\n");
        return -1;
    }
}

//设置读出模式，参数取值范围为 0~readModeNum-1，这里以模式 0 为例进行设置
retVal = SetQHYCCDReadMode(camHandle,0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Get Read Mode name successfully.\n");
}
else
{
    printf("Get Read Mode name failed.\n");
    return -1;
}
```

3.设置单帧或连续模式

设置相机工作在单帧或连续模式下。

获取相机句柄之后初始化之前可以设置此功能，使用的函数有 IsQHYCCDControlAvailable、SetQHYCCDStreamMode，下为设置单帧或连续模式的示例代码：

```
//设置单帧模式
retVal = IsQHYCCDControlAvailable(camHandle, CAM_SINGLEFRAMEMODE); //检查是否支持单帧模式
if(retVal == QHYCCD_SUCCESS)
{
    printf("This camera has single mode.\n");
}
else
{
    printf("This camera have no single mode.\n");
    return -1;
}
```

```
retVal = SetQHYCCDStreamMode(camHandle, 0);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Set Stream Mode successfully.\n");
}
else
{
	printf("Set Stream Mode failed.\n");
	return -1;
}

//设置连续模式

retVal = IsQHYCCDControlAvailable(camHandle, CAM_LIVEFRAMEMODE); //检查是否支持连续模式
if(retVal == QHYCCD_SUCCESS)
{
	printf("This camera has single mode.\n");
}
else
{
	printf("This camera have no single mode.\n");
	return -1;
}

retVal = SetQHYCCDStreamMode(camHandle, 1);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Set Stream Mode successfully.\n");
}
else
{
	printf("Set Stream Mode failed.\n");
	return -1;
}
```

4.初始化相机

初始化相机的参数，调整相机的工作状态。

获取相机句柄之后可以执行此设置，使用的函数为 InitQHYCCD，下为初始化相机的示例代码：

```
retVal = InitQHYCCD(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Initialize camera successfully.\n");
}
else
{
	printf("Initialize camera successfully.\n");
	return -1;
}
```

5.连接相机

此操作为操作 1234 的总合，用于打开相机并对相机进行初始化设置，操作包含获取相机的设备句柄，通过句柄设置相机的读出模式和工作模式，并对相机参数进行初始化设置。

用到的函数有 InitQHYCCDResource、ScanQHYCCD、GetQHYCCDId、OpenQHYCCD、GetQHYCCDNumberOfReadMode、GetQHYCCDReadModeName、GetQHYCCDReadModeResolution、SetQHYCCDReadMode、SetQHYCCDStreamMode 和 InitQHYCCD，下为连接相机的示例代码：

```
int camNum = 0,readModeNum = 0;
uint32_t retVal = QHYCCD_ERROR;
char camId[40] = {0};
char readModeName[40] = {0};
int readModeWidth = 0,readModeHeight = 0;
qhyccd_handle *camHandle = NULL;

/*****************************************************************************/
/*************************** 打开相机 获取句柄 ***************************/
/*****************************************************************************/
//初始化 SDK 内部的资源
retVal = InitQHYCCDResource();
if(retVal == QHYCCD_SUCCESS)
{
	printf("Initialize SDK resource successfully.\n");
}
else
{
	printf("Initialize SDK resource failed.\n");
	return -1;
}

//扫描相机，并返回设备数量
camNum = ScanQHYCCD();
if(camNum == 0)
{
	printf("Didn't find QHYCCD cameras.\n");
	return -1;
}

//连接 QHY 相机列表里的第一个设备
for(int i = 0; i < camNum; i ++)
{
	retVal = GetQHYCCDId(i, camId); //获取设备 ID
	if(retVal == QHYCCD_SUCCESS)
	{
		printf("Get camera's ID successfully.\n");
		break;
```

```
	}
	else
	{
		printf("Get camera's ID failed.\n");
		return -1;
	}
}

camHandle = OpenQHYCCD(camId); //根据设备 ID 打开设备，并返回设备句柄
if(camHandle != NULL)
{
	printf("Openc camera successfully.\n");
}
else
{
	printf("Openc camera failed.\n");
	return -1;
}

/*****************************************************************************/
/*********************************  模式设置  ********************************/
/*****************************************************************************/
//获取相机读出模式的数量，返回 1 则说明只有一个标准读出模式（STANDARD MODE）
retVal = GetQHYCCDNumberOfReadMode(camHandle,readModeNum);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Get Read Mode number successfully.\n");
}
else
{
	printf("Get Read Mode number failed.\n");
	return -1;
}

for(int i = 0; i < readModeNum; i++)
{
	//获取读出模式的名称
	retVal = GetQHYCCDReadModeName(camHandle, i, readModeName);
	if(retVal == QHYCCD_SUCCESS)
	{
		printf("Get Read Mode name successfully.\n");
	}
	else
	{
		printf("Get Read Mode name failed.\n");
		return -1;
	}

	//获取当前读出模式的分辨率
	retVal = GetQHYCCDReadModeResolution(camHandle, i, readModeWidth,readModeHeight);
	if(retVal == QHYCCD_SUCCESS)
	{
```

```
            printf("Get Read Mode resolution successfully.\n");
        }
        else
        {
            printf("Get Read Mode resolution failed.\n");
            return -1;
        }
}

//设置读出模式，参数取值范围为 0~readModeNum-1，这里以模式 0 为例进行设置
retVal = SetQHYCCDReadMode(camHandle,0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Get Read Mode name successfully.\n");
}
else
{
    printf("Get Read Mode name failed.\n");
    return -1;
}

//设置单帧或连续模式，单帧参数为 0 连续则为 1，这里设置的是单帧模式
retVal = IsQHYCCDControlAvailable(camHandle, CAM_SINGLEFRAMEMODE); //检查是否支持单帧模式
if(retVal == QHYCCD_SUCCESS)
{
    printf("This camera has single mode.\n");
}
else
{
    printf("This camera have no single mode.\n");
    return -1;
}

retVal = SetQHYCCDStreamMode(camHandle, 0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set Stream Mode successfully.\n");
}
else
{
    printf("Set Stream Mode failed.\n");
    return -1;
}

/*********************************************************************************/
/********************************* 初始化相机 ************************************/
/*********************************************************************************/
retVal = InitQHYCCD(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Initialize camera successfully.\n");
}
```

```
else
{
    printf("Initialize camera successfully.\n");
    return -1;
}
```

6.断开相机

关闭相机并释放相关资源，断开相机时有两个需要注意的地方，一是断开相机之前，若相机正在拍摄需要先结束拍摄任务，具体操作请查看结束拍摄任务功能介绍；二是执行完 CloseQHYCCD 函数之后，相机的设备句柄将被销毁，此时请勿与相机进行交互操作。

用到的函数有 CloseQHYCCD 和 ReleaseQHYCCDResource，具体代码如下：

```
//关闭相机
retVal = CloseQHYCCD(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Close camera successfully.\n");
}
else
{
    printf("Close camera failed.\n");
}

//释放 SDK 资源
retVal = ReleaseQHYCCDResource();
if(retVal == QHYCCD_SUCCESS)
{
    printf("Release resource successfully.\n");
}
else
{
    printf("Release resource failed.\n");
}
```

7.检查相机支持的功能

根据 CONTROL_ID 检查相机是否支持某项功能，获取相机的设备句柄之后即可使用此功能。

通过 OpenQHYCCD 函数获取相机句柄之后即可使用此功能，使用的函数为 IsQHYCCDControlAvailable，以亮度功能为例，检查相机是否支持此功能的代码为：

```
int retVal = QHYCCD_ERROR;

retVal = IsQHYCCDControlAvailable(camhandle, CONTROL_BRIGHTNESS);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Camera have this function.\n");
}
else
{
    printf("Camera don't have this function.\n");
}
```

CONTROL_ID 的定义位于 qhyccdstruct.h 中，下为所有 CONTROL_ID 的定义：

序号	CONTROL_ID	说明
0	CONTROL_BRIGHTNESS	检查相机是否支持亮度设置功能
1	CONTROL_CONTRAST	检查相机是否支持对比度设置功能
2	CONTROL_WBR	检查相机是否支持红色白平衡设置功能
3	CONTROL_WBB	检查相机是否支持蓝色白平衡设置功能
4	CONTROL_WBG	检查相机是否支持绿色白平衡设置功能
5	CONTROL_GAMMA	检查相机是否支持 Gamma 设置功能
6	CONTROL_GAIN	检查相机是否支持增益设置功能
7	CONTROL_OFFSET	检查相机是否支持偏置设置功能
8	CONTROL_EXPOSURE	检查相机是否支持曝光时间设置功能
9	CONTROL_SPEED	检查相机是否支持速度设置功能
10	CONTROL_TRANSFERBIT	检查相机是否支持图像位数设置功能
11	CONTROL_CHANNELS	检查相机是否可以检测通道数，目前已停用
12	CONTROL_USBTRAFFIC	检查相机是否支持设置 USB Traffic 功能
13	CONTROL_ROWNOISERE	检查相机是否支持行降噪功能
14	CONTROL_CURTEMP	检查相机是否可以获取当前温度
15	CONTROL_CURPWM	检查相机是否可以获取当前制冷功率
16	CONTROL_MANULPWM	检查相机是否可以支持手动制冷功能
17	CONTROL_CFWPORT	检查相机是否支持滤镜轮控制功能
18	CONTROL_COOLER	检查相机是否支持自动制冷功能
19	CONTROL_ST4PORT	检查相机是否支持 ST4 导星接口
20	CAM_COLOR	检查相机是否可以查看 Bayer 序列
21	CAM_BIN1X1MODE	检查相机是否可以设置 1X1 BIN
22	CAM_BIN2X2MODE	检查相机是否可以设置 2X2 BIN
23	CAM_BIN3X3MODE	检查相机是否可以设置 3X3 BIN
24	CAM_BIN4X4MODE	检查相机是否可以设置 4X4 BIN
25	CAM_MECHANICALSHUTTER	检查相机是否支持机械快门
26	CAM_TRIGER_INTERFACE	检查相机是否支持外触发功能
27	CAM_TECOVERPROTECT_INTERFACE	检查相机是否支持制冷器过温保护功能，此功能会限制制冷器功率上限为 70%（未启用）
28	CAM_SINGNALCLAMP_INTERFACE	检查相机是否支持 SINGNALCLAMP 功能，此功能为 CCD 相机特有的功能，针对亮星后的暗带
29	CAM_FINETONE_INTERFACE	检查相机是否支持微调功能，此功能针对 CCD 相机，可以通过对 CCD 的驱动和采样时序微调，来优化相机的噪声特性
30	CAM_SHUTTERMOTORHEATING_INTERFACE	检查相机是否支持快门电机加热功能
31	CAM_CALIBRATEFPN_INTERFACE	检查相机是否支持 FPN 校准功能，此功能用来降低 FPN 噪声，如垂直条纹
32	CAM_CHIPTEMPERATURESENSOR_INTERFACE	检查相机是否支持片上温度传感器
33	CAM_USBREADOUTSLOWEST_INTERFACE	检查相机是否支持 USB 最低速读出功能（此功能和 CONTROL_SPEED 功能重复，已不再使用）
34	CAM_8BITS	检查相机是否支持 8 位图像数据输出功能
35	CAM_16BITS	检查相机是否支持 16 位图像数据输出功能
36	CAM_GPS	检查相机是否支持 GPS 功能
37	CAM_IGNOREOVERSCAN_INTERFACE	检查相机是否支持过扫区校准功能
38	QHYCCD_3A_AUTOBALANCE	检查相机是否支持自动白平衡功能
39	QHYCCD_3A_AUTOEXPOSURE	检查相机是否支持自动曝光功能
40	QHYCCD_3A_AUTOFOCUS	检查相机是否支持自动调焦功能

41	CONTROL_AMPV	检查相机是否支持辉光抑制功能
42	CONTROL_VCAM	检查相机是否支持 WDM 广播功能
43	CAM_VIEW_MODE	检查是否支持预览模式（未启用）
44	CONTROL_CFWSLOTSNUM	检查相机是否可以获取滤镜轮孔数
45	IS_EXPOSING_DONE	检查相机是否曝光完成（未启用）
46	ScreenStretchB	检查相机是否可以 Black 灰度拉伸
47	ScreenStretchW	检查相机是否可以 White 灰度拉伸
48	CONTROL_DDR	检查相机是否支持 DDR 功能
49	CAM_LIGHT_PERFORMANCE_MODE	检查相机是否支持高低增益切换功能
50	CAM_QHY5II_GUIDE_MODE	检查相机是否是支持导星模式的 5II 系列相机
51	DDR_BUFFER_CAPACITY	检查相机是否可以获取当前 DDR 缓冲区数据量
52	DDR_BUFFER_READ_THRESHOLD	检查相机是否可以获取缓冲区读阈出值
53	DefaultGain	检查相机是否可以获取默认增益推荐值
54	DefaultOffset	检查相机是否可以获取默认偏置推荐值
55	OutputDataActualBits	检查相机是否可以获取输出数据实际位数
56	OutputDataAlignment	检查相机是否支持获取输出数据对齐格式
57	CAM_SINGLEFRAMEMODE	检查相机是否支持单帧模式
58	CAM_LIVEVIDEOMODE	检查相机是否支持连续模式
59	CAM_IS_COLOR	检查相机是否是彩色相机
60	hasHardwareFrameCounter	检查相机是否支持硬件帧计数功能
61	CONTROL_MAX_ID_Error	获取 CONTROL_ID 的最大值（已弃用）
62	CAM_HUMIDITY	检查相机是否支持湿度传感器
63	CAM_PRESSURE	检查相机是否支持压力传感器
64	CONTROL_VACUUM_PUMP	检查相机是否支持真空泵
65	CONTROL_SensorChamberCycle_PUMP	检查相机是否支持内部循环泵
66	CAM_32BITS	检查相机是否支持 32 位图像数据输出功能
67	CAM_Sensor_ULVO_Status	检查相机是否支持 ULVO 状态检测功能
68	CAM_SensorPhaseReTrain	检查相机是否支持调整相位功能，此功能可处理因相位导致的图像条纹问题
69	CAM_InitConfigFromFlash	检查相机是否支持 Flash 读写 config 功能
70	CAM_TRIGER_MODE	检查相机是否支持多种触发模式设置功能
71	CAM_TRIGER_OUT	检查相机是否支持触发输出功能
72	CAM_BURST_MODE	检查相机是否支持 Burst 模式
73	CAM_SPEAKER_LED_ALARM	检查相机是否支持信号灯功能（目前仅针对定制型号）
74	CAM_WATCH_DOG_FPGA	检查相机 FPGA 是否支持看门狗处理功能（目前仅针对定制型号）
75	CAM_BIN6X6MODE	检查相机是否支持 6X6 BIN
76	CAM_BIN8X8MODE	检查相机是否支持 8X8 BIN
77	CAM_GlobalSensorGPSLED	检查相机传感器是否支持全局 LED 校准灯
78	CONTROL_ImgProc	检查相机是否支持图像处理功能
79	CONTROL_RemoveRBI	检查相机是否具有消除残影功能
80	CONTROL_GlobalReset	检查相机是否具有全局复位功能
81	CONTROL_FrameDetect	检查相机是否具有帧监测功能
82	CAM_GainDBConversion	检查相机是否具有增益转换功能
83	CAM_CurveSystemGain	检查相机是否具有获取系统增益曲线功能
84	CAM_CurveFullWell	检查相机是否具有获取满井曲线功能
85	CAM_CurveReadoutNoise	检查相机是否具有获取读出噪声曲线功能
86	CAM_UseAverageBinning	检查相机是否具有启用均值合并功能

.....		
-	CONTROL_MAX_ID	获取可用的 CONTROL_ID 的最大值
以下功能为测试功能，可能会有使用上或者稳定性的问题		
1024	CONTROL_AUTOWHITEBALANCE	检查相机是否具有自动白平衡功能
1025	CONTROL_AUTOEXPOSURE	检查相机是否具有自动曝光功能
1026	CONTROL_AUTOEXPmessureValue	检查相机是否具有设置自动曝光阈值功能
1027	CONTROL_AUTOEXPmessureMethod	检查相机是否具有设置自动曝光遮罩模式功能
1028	CONTROL_ImageStabilization	检查相机是否具有稳像功能
1029	CONTROL_GAINdB	检测相机是否具有设置 dB 增益功能
1030	CONTROL_DPC	检测相机是否具有去除热噪声功能
1031	CONTROL_DPC_value	检测相机是否具有设置热噪声阈值功能

8.获取 SDK 的版本号

获取当前 SDK 使用的版本号，通常版本号为发布 SDK 的日期。

任何时候都可以使用此功能，使用的函数为 GetQHYCCDSDKVersion，代码如下：

```
int retVal = QHYCCD_ERROR;
uint32_t year,month,day,subday;

retVal = GetQHYCCDSDKVersion(&year,&month,&day,&subday);
if(retVal == QHYCCD_SUCCESS)
{
    printf("The SDK version is %d-%d-%d-%d.\n", year, month, day, subday);
}
```

9.获取驱动的版本号

获取相机固件（firmware）的版本，版本通常为发布日期或修改日期。

通过 OpenQHYCCD 函数获取设备句柄后即可使用此功能，使用的函数为 GetQHYCCDFWVersion，示例代码如下：

```
int retVal = QHYCCD_ERROR;
unsigned char fwVer[32];

retVal = GetQHYCCDFWVersion(camHandle,fwVer);
if(retVal == QHYCCD_SUCCESS)
{
    if((fwv[0] >> 4) <= 9)
    {
        printf("Firmware Version: %d-%d-%d", (fwVer[0]>>4)+0x10, fwVer[0]&~0xf0, fwVer[1]);
            }
    else
    {
        printf("Firmware Version: %d-%d-%d", fwVer[0]>>4, fwVer[0]&~0xf0, fwVer[1]);
    }
}
```

10.获取当前设备 FPGA 版本号

获取设备的 FPGA 版本号，设备的 FPGA 版本号有两个，第一个是普通的版本号，另一个为备用的版本号，一般只需要读取第一个版本号。

通过 OpenQHYCCD 函数获取设备句柄后即可使用此功能，使用的函数为 GetQHYCCDFPGAVersion，示例

代码如下：

```
int retVal = QHYCCD_ERROR;
uint8_t fpgaVer[32] = {0};

//获取第一个 FPGA 版本号
retVal = GetQHYCCDFPGAVersion(camHandle,0,fpgaVer);
if(retVal == QHYCCD_SUCCESS)
{
    printf("First FPGA Version: %d-%d-%d-%d", fpgaVer[0],fpgaVer[1],fpgaVer[2],fpgaVer[3]);
}

//获取第二个 FPGA 版本号
retVal = GetQHYCCDFPGAVersion(camHandle,1,fpgaVer);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Second FPGA Version: %d-%d-%d-%d", fpgaVer[0],fpgaVer[1],fpgaVer[2],fpgaVer[3]);
}
```

11.获取相机芯片信息

获取相机的硬件参数，包括芯片尺寸、像素尺寸以及图像尺寸和图像位深，需要注意的是，图像的位深是不固定的，当设置位深后，由此函数得到的图像位深也会发生变化。

通过 InitQHYCCD 函数初始化之后可以使用此功能，使用的函数为 GetQHYCCDChipInfo，示例代码如下：

```
uint32_t retVal = QHYCCD_SUCCESS;
uint32_t imagew, imageh, bpp;
double chipw, chiph, pixelw, pixelh;

retVal = GetQHYCCDChipInfo(camHandle, &chipw, &chiph, &imagew, &imageh, &pixelw, &pixelh, &bpp);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Chip Width: %f mm Chip Height: %f mm", chipw, chiph);
    printf("Pixel Width: %f nm Pixel Height: %f nm", pixelw, pixelh);
    printf("Image Width: %d Image Height: %d Image Bits: %d", imagew,imageh,bpp);
}
```

12.设置 BIN 模式

通常相机有四种 BIN 模式，分别是 1X1 BIN、2X2 BIN、3X3 BIN 和 4X4 BIN，不同相机支持的 BIN 模式不同，因此设置之前需要先检查相机是否支持 BIN 模式设置，另外，设置的同时需要一起设置分辨率，此功能为必要功能设置。

通过 InitQHYCCD 函数初始化之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDChipInfo、SetQHYCCDBinMode 和 SetQHYCCDResolution，示例代码如下：

```
//设置 1x1 BIN
int retVal = QHYCCD_ERROR;
uint32_t binX = 1, binY = 1;
uint32_t imgW, imgH, imgBpp;
double chipW, chipH, pixelW, pixelH;

//检测是否支持 1x1 BIN 功能
```

```
retVal = IsQHYCCDControlAvailable(camHandle, CAM_BIN1X1MODE);
if(retVal == QHYCCD_SUCCESS)
{
    //设置 1x1BIN
    retVal = SetQHYCCDBinMode(camHandle, binX, binY);
    if(retVal == QHYCCD_SUCCESS)
    {
        retVal = GetQHYCCDChipInfo(camHandle,&chipW,&chipH,&imgW,&imgH,&pixelW,&pixelH,&imgBpp);
        if(retVal == QHYCCD_SUCCESS)
        {
            //设置 BIN 的同时需要设置分辨率
            retVal = SetQHYCCDResolution(camHandle, 0, 0, imgW / binX, imgH / binY);
            if(retVal == QHYCCD_SUCCESS)
            {
                printf("Set Resolution Successfully.\n");
            }
        }
    }
}
```

13.设置彩色模式并获取 Bayer 序列

设置彩色模式开启或关闭，并获取相机的 Bayer 序列，此功能为必要设置。只有彩色相机支持此功能设置，设置前需先调用 IsQHYCCDControlAvailable 函数检查相机是否是彩色相机。

通过 InitQHYCCD 函数初始化之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable 和 SetQHYCCDDebayerOnOff，示例代码如下：

```
uint32_t retVal = QHYCCD_SUCCESS;
int bayer;

//检测是否是彩色相机
retVal = IsQHYCCDConrolAvail(camHandle, CAM_IS_COLOR);
if(retVal == QHYCCD_SUCCESS)
{
    //开启彩色模式
    retVal = SetQHYCCDDebayerOnOff(camhandle, true);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Setup Color Mode ON Successfully.\n");
    }

    //获取相机 Bayer 序列
    bayer = IsQHYCCDControlAvailable(camhandle, CAM_COLOR);
    if(retVal != QHYCCD_ERROR)
    {
        printf("Get camera's bayer format successfully.\n");
    }
}
```

关于相机的 Bayer 序列，这个是定义在 qhyccdstruct.h 中的四个枚举变量，枚举定义如下：

```
enum BAYER_ID
{
  BAYER_GB = 1,
```

```
  BAYER_GR,
  BAYER_BG,
  BAYER_RG
};
```

这四个枚举变量分别代表 GBRG、GRBG、BGGR 和 RGGB。

14.设置位数及图像数据格式

设置输出的图像数据位数，此功能为必要功能设置。注意，输出的图像数据位数和原始数据位数可能不一致，如相机的原始数据为 12 位，此时若设置相机输出 16 位数据，相机内部会在原始数据的低位补零，转换成十六位数据，如果设置相机输出 8 位数据，相机内部会取原始数据的高八位，转换成八位数据。

使用 InitQHYCCD 函数初始化之后可以使用此功能，使用的函数为 SetQHYCCDParam 或 SetQHYCCDBitsMode，这两个函数的功能相同，下为设置图像位数的示例代码：

```
//设置 8 位图像
uint32_t retVal = QHYCCD_ERROR;
retVal = IsQHYCCDControlAvailable(camHandle, CAM_8BITS);
if(retVal == QHYCCD_SUCCESS)
{
    //retVal = SetQHYCCDBitsMode(camhandle, 8); //与 SetQHYCCDParam 具有相同功能
    retVal = SetQHYCCDParam(camHandle, CONTROL_TRANSFERBIT, 8);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set Image Bits Successfully.\n");
    }
}
//设置 16 位图像
uint32_t retVal = QHYCCD_ERROR;
retVal = IsQHYCCDControlAvailable(camHandle, CAM_8BITS);
if(retVal == QHYCCD_SUCCESS)
{
    //retVal = SetQHYCCDBitsMode(camhandle, 16); //与 SetQHYCCDParam 具有相同功能
    retVal = SetQHYCCDParam(camHandle, CONTROL_TRANSFERBIT, 16);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set Image bits successfully.\n");
    }
}
```

此函数一般与 SetQHYCCDDebayerOnOff 函数一起使用，用来设置图像的数据格式，设置图像的数据结构时会因单帧和连续模式的不同而有所差异。

在单帧模式下，有些相机无法输出八位图像，并且天文软件在使用单帧模式拍摄时一般选用十六位黑白模式，因此单帧模式下一般将相机统一设置成十六位黑白模式，此时不需要考虑数据格式切换的问题。如果在连续模式下工作，通常有八位黑白（RAW8）、八位彩色（RGB24）和十六位黑白（RAW16）三种图像数据模式，此时需要考虑数据格式切换的问题，详情请参考读出模式、BIN、数据格式切换功能。

下为设置三种数据格式的示例代码：

```
//设置八位黑白模式
retVal = SetQHYCCDParam(camHandle, CONTROL_TRANSFERBIT, 8);
if(retVal == QHYCCD_SUCCESS)
```

```
{
    printf("Set bits mode successfully.\n");
}
retVal = SetQHYCCDDebayerOnOff(camHandle, false);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set color mode successfully.\n");
}

//设置 16 位黑白模式
retVal = SetQHYCCDParam(camHandle, CONTROL_TRANSFERBIT, 16);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set bits mode successfully.\n");
}
retVal = SetQHYCCDDebayerOnOff(camHandle, false);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set color mode successfully.\n");
}

//设置八位彩色模式
retVal = SetQHYCCDParam(camHandle, CONTROL_TRANSFERBIT, 8);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set bits mode successfully.\n");
}
retVal = SetQHYCCDDebayerOnOff(camHandle, true);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set color mode successfully.\n");
}
```

15.切换读出模式、BIN 模式、图像数据格式

当拍摄途中需要切换相机的读出模式、BIN 模式或数据格式时，需要先结束相机的拍摄任务，然后再对读出模式、BIN 模式、数据格式等进行设置，设置时需要调用 InitHYCCD 函数初始化相机，设置完成后继续开始拍摄任务。另外需要注意的是，由于切换时执行了初始化操作，因此切换完成后按照之前的参数值需要重新，如 Expose Time、Gain、Offset、Traffic 等参数。

连接相机后可以使用切换功能，可以重复多次切换，使用的函数有 SetQHYCCDReadMode、SetQHYCCDStreamMode、InitQHYCCD、SetQHYCCDBinMode、SetQHYCCDResolution、SetQHYCCDParam 等，示例代码如下：

```
//结束相机拍摄任务，详情请查看单帧或连续模式拍摄功能

//重新设置读出模式
retVal = SetQHYCCDReadMode(camHandle, readMode);
if(retVal == QHYCCD_ERROR){
    printf("Get Read Mode name failed.\n");
    return -1;
}
```

```
//重新设置 Stream Mode
retVal = SetQHYCCDStreamMode(camHandle, streamMode);
if(retVal == QHYCCD_ERROR){
	printf("Set Stream Mode failed.\n");
	return -1;
}

//重新初始化相机
retVal = InitQHYCCD(camHandle);
if(retVal == QHYCCD_ERROR){
	printf("Initialize camera successfully.\n");
	return -1;
}

//重新设置 BIN 模式
retVal = SetQHYCCDBinMode(camhandle, binx, biny);
if(retVal == QHYCCD_ERROR){
	printf("Set Bin Mode failed.\n");
	return -1;
}
retVal = SetQHYCCDResolution(camhandle, startx, starty, sizex, sizey);
if(retVal == QHYCCD_ERROR){
	printf("Set Resolution failed.\n");
	return -1;
}

//重新设置数据格式
retVal = SetQHYCCDDebayerOnOff(camhandle, color);
if(retVal == QHYCCD_ERROR){
	printf("Set Color Mode failed.\n");
	return -1;
}
retVal = SetQHYCCDParam(camhandle, CONTROL_TRANSFERBIT, bits);
if(retVal == QHYCCD_ERROR){
	printf("Set Bits Mode failed.\n");
	return -1;
}

//重新设置相机参数，详情请查看参数设置功能
retVal = SetQHYCCDParam(camhandle, CONTROL_EXPOSURE, time);
if(retVal == SetQHYCCD_SUCCESS)
{
	printf("Setup expose time successfully.\n");
}
retVal = SetQHYCCDParam(camhandle, CONTROL_GAIN, gain);
if(retVal == SetQHYCCD_SUCCESS)
{
	printf("Setup gain successfully.\n");
}
retVal = SetQHYCCDParam(camhandle, CONTROL_OFFSET, offset);
if(retVal == SetQHYCCD_SUCCESS)
```

```
{
    printf("Setup offset successfully.\n");
}
retVal = SetQHYCCDParam(camhandle, CONTROL_USBTRAFFIC, traffic);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup traffic successfully.\n");
}
...

//重新开始拍摄任务，详情请查看单帧或连续模式拍摄功能
```

16.设置和获取 ROI（region of interest）

以左上角第一个像素点为基准点，显示指定区域的图像。

ROI 有两种，分别是软件 ROI 和硬件 ROI，软件 ROI 是相机仍然输出整幅图像，SDK 内部对图像进行裁剪，因此图像的读出时间并不会缩短，也无法提高连续模式下的帧率，硬件 ROI 是指在相机内部进行图像裁剪，只输出指定区域的图像数据，因此会缩短图像的读出时间，提高连续模式下的帧率。

设置 BIN 模式之后可以使用此功能，此功能为可选功能，使用的函数为 SetQHYCCDResolution、GetQHYCCDCurrentROI，下为从左上角第一个像素点开始，显示尺寸为 500x400 的图像的示例代码：

```
//设置 ROI
int retVal = QHYCCD_ERROR;

retVal = SetQHYCCDResolution(camHandle, 0, 0, 500, 400);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set ROI Successfully.\n");
}

//获取 ROI 设置
uint32_t x, y, sizex, sizey;
retVal = GetQHYCCDCurrentROI(camHandle, &x, &y, &sizex, &sizey);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Get ROI setting successfully.\n");
}
```

17.获取相机过扫区范围

获取图像的过扫区范围，做暗场校正时需要用到过扫区的图像数据，若返回的过扫区范围参数为 0，则说明该相机没有过扫区，不同 BIN 模式下过扫区的起始位置和尺寸不同。

设置 BIN 模式之后可以使用此功能，使用的函数为 GetQHYCCDOverScanArea，下为获取过扫区范围的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
uint32_t startx, starty, sizex, sizey;

retVal = GetQHYCCDOverScanArea(camHandle, &startx, &starty, &sizex, &sizey);
if(retVal == QHYCCD_SUCCESS)
{
```

```
    printf("Get overscan area successfully.\n");
}
```

18.获取相机有效区域范围

获取相机的有效区域范围，根据此范围可以设置 ROI，直接输出有效区域。

设置 BIN 模式后可以使用此功能，使用的函数为 GetQHYCCDEffectiveArea，下为获取有效区域的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
uint32_t startx, starty, sizex, sizey;

retVal = GetQHYCCDEffectiveArea(camHandle, &startx, &starty, &sizex, &sizey);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Get effective area successfully.\n");
}
```

19.设置相机过扫区矫正

设置相机的过扫区校正，设置之后将移除过扫区，只输出有效区域的图像数据。

设置过扫区校正有两种方式，第一种方式是通过设置 ROI 实现的，需要先获取有效区域范围，然后通过 SetQHYCCDResolution 函数设置 ROI，直接输出有效区域的图像数据。

设置 BIN 模式之后可以使用此功能，也可以在设置 BIN 模式的时候直接设置，使用的函数有 GetQHYCCDEffectiveArea 和 SetQHYCCDResolution，示例代码如下：

```
uint32_t startx, starty, sizex, sizey;

retVal = GetQHYCCDEffectiveArea(camHandle, &startx, &starty, &sizex, &sizey);
if(retVal == QHYCCD_SUCCESS)
{
    retVal = SetQHYCCDResolution(camhandle, startx, starty, sizex, sizey);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set ROI successfully.\n");
    }
}
```

第二种方式是通过 SetQHYCCDParam 函数进行设置，设置完成后将只输出有效区域的图像数据，两种设置方式的的区别在于，当同时设置过扫区校正和 ROI 时，ROI 的参考点会产生变化，具体会在下面的 BIN、ROI、过扫区混合使用功能中介绍。

设置 BIN 模式之后可以使用此功能，下为设置过扫区校正的示例代码：

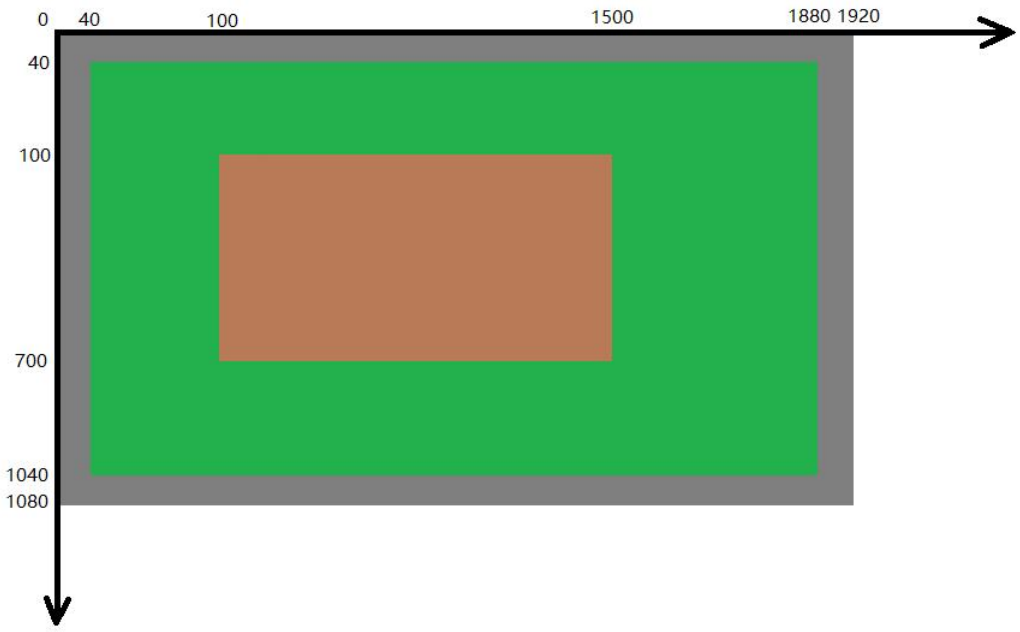
```
retVal = SetQHYCCDParam(camHandle, CAM_IGNOREOVERSCAN_INTERFACE, 1.0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set ignore overscan successfully.\n");
}
```

另外需要注意的是，这两种方式的实现方式是有冲突的，不能同时使用，并且建议使用第一种设置方式。

20.BIN、ROI 和过扫区校正的混合使用

当使用方式一设置过扫区校正时，相机所有关于图像裁剪的操作，都是以原始图像左上角的第一个像素为基准点进行设置，若将图像比作坐标系，则左上角第一个像素为坐标原点。

假设一张全分辨率尺寸为 1920x1080 的图像，其过扫区尺寸为宽 40 高 40，分布于图像四周，有效区域起始位置为(40,40)，尺寸为 1840x1000，ROI 起始位置为(100,100)，尺寸为 1400x600，如下图所示：



此时设置 ROI 的代码如下：

```
retVal = SetQHYCCDResolution(camHandle, 100, 100, 1400, 600);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

设置过扫区校正的代码如下：

```
retVal = SetQHYCCDResolution(camHandle, 40, 40, 1840, 1000);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

当设置过扫区校正之后，图像将只显示有效区域部分，ROI 的起始位置将从(100, 100)变成(60, 60)，但实际上设置 ROI 仍以原来的全分辨率的图像为基准进行设置，设置时起始位置需要加上过扫区的尺寸，此时若设置同样位置和尺寸的 ROI，其代码为：

```
retVal = SetQHYCCDResolution(camHandle, 40 + 60, 40 + 60, 1400, 600);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

设置 2X2 BIN 之后的过扫区校正和 ROI 与 1X1 BIN 模式下类似，不过尺寸变为原来的一半，例如上面示例图经过 2X2 BIN 之后就会变成全分辨率尺寸为 960x540 的图像，其过扫区尺寸为宽 20 高 20，有效区域起始位置为(20,20)，尺寸为 920x500，ROI 起始位置为(50,50)，尺寸为 700x300。

此时设置 ROI 的代码为：

```
retVal = SetQHYCCDResolution(camHandle, 50, 50, 700, 300);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

设置过扫区校正的代码为：

```
retVal = SetQHYCCDResolution(camHandle, 20, 20, 920, 500);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

和 1X1 BIN 模式类似，设置过扫区校正之后，图像将只显示有效区域部分，此时从图像上来看，ROI 起始位置从原来的(50,50)变成了(30,30)。但实际上设置 ROI 仍以 2X2 BIN 模式下的全分辨率的图像为基准进行设置，设置时起始位置需要加上过扫区的尺寸，此时若设置同样位置和尺寸的 ROI，其代码为：

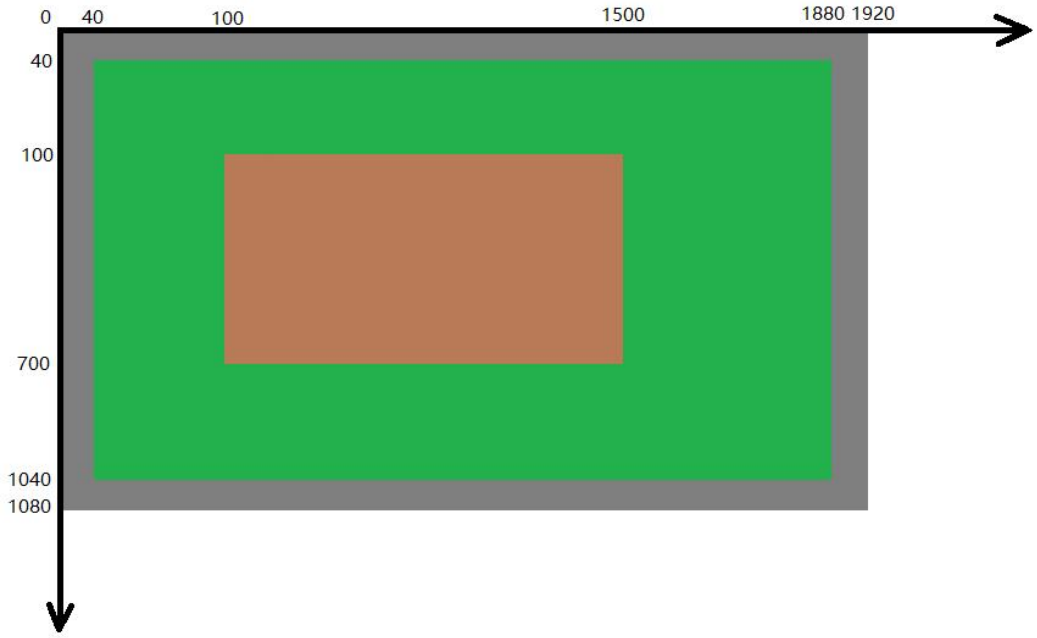
```
retVal = SetQHYCCDResolution(camHandle, 20 + 30, 20 + 30, 920, 500);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

其他 BIN 模式的原理也是相同的，可以使用同样方法进行设置。

方式二：

当使用方式二设置过扫区校正时，ROI 的参考位置将发生变化，其参考位置将变成有效区域左上角第一个像素点。

假设一张全分辨率尺寸为 1920x1080 的图像，其过扫区尺寸为宽 40 高 40，分布于图像四周，有效区域起始位置为(40,40)，尺寸为 1840x1000，ROI 起始位置为(100,100)，尺寸为 1400x600，如下图所示：



此时设置 ROI 的代码如下：

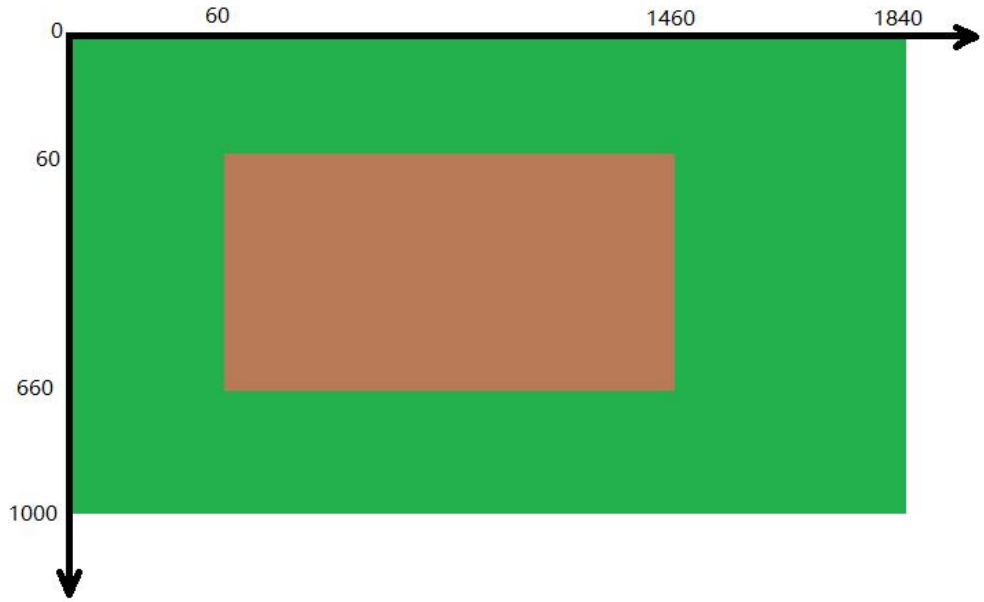
```
retVal = SetQHYCCDResolution(camHandle, 100, 100, 1400, 600);
if(retVal == SetQHYCCD_SUCCESS)
{
```

```
    printf("Setup ROI successfully.\n");
}
```

设置过扫区校正的代码如下：

```
retVal = SetQHYCCDParam(camHandle, CAM_IGNOREOVERSCAN_INTERFACE, 1.0);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

当设置过扫区校正之后，图像将只显示有效区域部分，ROI 的起始位置将从(100, 100)变成(60, 60)，如下图所示：



此时若设置同样位置和尺寸的 ROI，其代码为：

```
retVal = SetQHYCCDResolution(camHandle, 60, 60, 1400, 600);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

设置 2X2 BIN 之后的过扫区校正和 ROI 与 1X1 BIN 模式下类似，不过尺寸变为原来的一半，例如上面示例图经过 2X2 BIN 之后就会变成全分辨率尺寸为 960x540 的图像，其过扫区尺寸为宽 20 高 20，有效区域起始位置为(20,20)，尺寸为 920x500，ROI 起始位置为(50,50)，尺寸为 700x300。

此时设置 ROI 的代码为：

```
retVal = SetQHYCCDResolution(camHandle, 50, 50, 700, 300);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

设置过扫区校正的代码为：

```
retVal = SetQHYCCDResolution(camHandle, 20, 20, 920, 500);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
```

```
}
```

和 1X1 BIN 模式类似，设置过扫区校正之后，图像将只显示有效区域部分，ROI 起始位置从原来的(50,50)变成了(30,30)。此时若设置同样位置和尺寸的 ROI，其代码为：

```
retVal = SetQHYCCDResolution(camHandle, 30, 30, 920, 500);
if(retVal == SetQHYCCD_SUCCESS)
{
    printf("Setup ROI successfully.\n");
}
```

其他 BIN 模式的原理也是类似的，可以使用同样方法进行设置。

21.获取数据输出的实际位数

获取相机实际输出的数据位数，此位数为芯片输出的原始数据的实际位数。相机内部会对原始数据进行处理，通过取高位的方式得到八位图像数据，或通过低位补零的方式得到十六位的图像数据。

设置数据格式后可以使用此功能，使用的函数为 GetQHYCCDParam，下为获取实际位数的示例代码：

```
uint32_t retVal;
retVal = (uint32_t)GetQHYCCDParam(camHandle, OutputDataActualBits);
if(retVal != QHYCCD_ERRROR)
{
    printf("Camera actual output bits is %d\n", retVal);
}
```

22.获取相机输出数据对齐格式

获取相机输出数据对齐格式，若返回值为 1，则说明为高位对齐，若返回值为 0，则说明为低位对齐。

```
uint32_t retVal;
retVal = (uint32_t)GetQHYCCDParam(camHandle, OutputDataAlignment);
if(retVal != QHYCCD_ERRROR)
{
    printf("Get camera output data aligenment successfully.\n");
}
```

23.获取相机图像所需的内存长度

获取相机图像所需的最大内存长度，此函数返回的内存长度为固定值，且会比实际所需值稍大一些以避免一些意外导致的内存溢出问题，其计算方式为:

黑白相机：(imagew+100)*(imageh+100)*2

彩色相机：(imagew+100)*(imageh+100)*3

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数为 GetQHYCCDMemLength，下为获取内存长度的示例代码：

```
int length = GetQHYCCDMemLength(camhandle);
if(length > 0)
{
    printf("Get image memory length successfully.\n");
```

```
}
```

另外，也可以根据图像尺寸及数据格式自行计算所需的内存长度，计算方式为：

```
length = imagew * imageh * channels * bits / 8
```

imagew 和 imageh 分别为设置图像分辨率的图像宽度和图像高度，channels 取决于彩色模式开启或关闭，当彩色模式开启时，channels 值为 3，当彩色模式关闭时，channels 值为 1，bits 为设置的图像数据位数。

24.设置和获取曝光时间

设置相机的曝光时间，曝光时间以微妙为单位，设置之前需要用 IsQHYCCDControlAvailable 函数检查是否可以设置曝光时间，可以使用 GetQHYCCDParamMinMaxStep 获取参数设置范围，检查和参数范围只确认一次就可以。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep 和 SetQHYCCDParam，下为设置曝光时间的示例代码：

```
int retVal = QHYCCD_ERROR;
double min, max, step;

//查看是否支持设置曝光时间
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_EXPOSURE);
if(retVal == QHYCCD_SUCCESS)
{
    //获取曝光时间设置范围
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_EXPOSURE, &min, &max, &step);
    if(retVal = QHYCCD_SUCCESS)
    {
        printf("Get expose time range successfully.\n");
    }

    //设置曝光时间
    retVal = SetQHYCCDParam(camHandle, CONTROL_EXPOSURE, 1000.0);   //设置 1ms 曝光时间
    if(retVal == QHYCCD_SUCCESS)
    {
       printf("Set Exposure Successfully.\n");
    }

    //获取曝光时间
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_EXPOSURE);
}
```

需要注意的是，无论单帧模式还是连续模式都可以调整曝光时间设置，但是如果相机正在曝光，则需要先终止曝光再设置，然后重新开始拍摄。

单帧模式下需要先调用 CancelQHYCCDExposingAndReadout 函数终止曝光，然后调整曝光时间，之后再调用 ExpQHYCCDSingleFrame 函数重新开始拍摄。不过单帧模式拍摄时，每次只拍摄一帧图像，因此很少需要考虑曝光中途更改曝光时间的问题。下为单帧模式下设置曝光时间的示例代码：

```
//如果正在曝光需要执行此函数终止曝光，否则不需要执行
retVal = CancelQHYCCDExposingAndReadout(camHandle(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
```

```
    printf("Cancel expose successfully.\n");
}

//设置曝光时间
retVal = SetQHYCCDParam(camHandle, CONTROL_EXPOSURE, 20000.0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set expose time successfully.\n");
}

//如果正在曝光需要执行此函数重新开始曝光，否则不需要执行
retVal = ExpQHYCCDSingleFrame(camHandle);
if(retVal != QHYCCD_ERROR)
{
    printf("Begin single capture successfully.\n");
}
```

连续模式下，需要考虑长曝光和短曝光两种情况。短曝光时，帧率刷新较快，不会花费太多时间，因此可以直接设置，但长曝光模式下，由于曝光时间设置会在下一帧生效，会占用过多时间，因此设置曝光时间之后，需要先调用 StopQHYCCDLive 函数终止曝光，再调用 BeginQHYCCDLive 函数重新开始拍摄。长曝光和短曝光的分界点可以自行决定，1s 或 2s 或其他时间都可以。下为连续模式下进行长曝光拍摄时设置曝光时间的示例代码：

```
//设置曝光时间
retVal = SetQHYCCDParam(camHandle, CONTROL_EXPOSURE, 20000.0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set expose time successfully.\n");
}

//结束连续模式
retVal = StopQHYCCDLive(camHnadle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Stop live capture successfully.\n");
}
//重新开始连续模式
retVal = BeginQHYCCDLive(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Begin live capture successfully.\n");
}
```

25.设置和获取增益（Gain）

设置相机的增益，参数值越大，图像越亮。所谓增益即是进行乘法运算，相机曝光完成后用图像数据乘以一个系数。乘法运算用到的系数由芯片决定，各个相机的增益曲线和 dB 增益可以在产品界面查看，此运算不会导致计算的像素值超过上限，如 8 位图像最大像素值通常为 255,16 位图像最大像素值通常为 65535。设置增益前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，并使用 GetQHYCCDParamMinMaxStep 函数确定参数设置范围，检查和参数设置范围只需要确定一次。

由于增益是曝光完成后进行的数据处理，因此可以不必结束拍摄任务再重新开始拍摄。另外，由于单帧和连续模式的差异，个别相机单帧和连续模式下的参数设置范围不一致。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep 和 SetQHYCCDParam，下为增益设置的示例代码：

```
int retVal = QHYCCD_ERROR;
double min, max, step;

//查看是否支持设置增益
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_GAIN);
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置的范围和最小步长
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_GAIN, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get gain range successfully.\n");
    }

    //设置增益
    retVal = SetQHYCCDParam(camHandle, CONTROL_GAIN, 1.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set gain successfully.\n");
    }

    //获取增益
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_OFFSET);
}
```

26.设置偏移（Offset）

设置相机的偏移值，参数值越大，图像越亮。和设置增益类似，偏移实际上也是对图像数据进行处理，不过区别在于，增益是乘法运算，偏移是加法运算，相机在曝光完成后在原有图像数据基础上再加上一定数值，此运算不会导致计算的像素值超过上限，如 8 位图像最大像素值通常为 255,16 位图像最大像素值通常为 65535。

同样的，设置偏移值时也可以直接设置，不必停止拍摄再重新开始，另外，个别相机的设置范围也会因单帧和连续模式而有所差异。

设置偏移值之前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，并使用 GetQHYCCDParamMinMaxStep 函数确定参数设置范围，检查和参数设置范围只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep 和 SetQHYCCDParam，下为设置偏移值的示例代码：

```
int retVal = QHYCCD_ERROR;
double min, max, step;

//查看是否支持设置偏移
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_OFFSET);
```

```
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置的取值范围和最小步长
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_OFFSET, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get offset range successfully.\n");
    }

    //设置 Offset
    retVal = SetQHYCCDParam(camHandle, CONTROL_OFFSET, 1.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set offset successfully.\n");
    }

    //获取偏移值
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_OFFSET);
}
```

27.设置和获取 Traffic

设置此参数可以调节连续模式下的帧率，以得到适合电脑的最大帧率，参数值越大，帧率帧率越慢。需要注意的是，此参数并不能直接设置连续模式的帧率为某一固定值，仅仅起到调节的作用。

其工作原理为，在每行图像数据的后面增加一些多余的空数据，这样可以增加整幅图像数据的读出时间以降低帧率，单帧模式也可以设置，但效果不会很明显，因为单帧模式本身读出速度相较于连续模式慢很多，即使增加一点时间，效果也不是很明显。

设置 Traffic 前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，并使用 GetQHYCCDParamMinMaxStep 函数确定参数设置范围，检查和参数设置范围只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep 和 SetQHYCCDParam，下为设置 Traffic 的示例代码：

```
int retVal = QHYCCD_ERROR;
double min, max, step;

//查看是否支持设置 Traffic
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_USBTRAFFIC);
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置的取值范围和最小步长
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_USBTRAFFIC, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get traffic range successfully.\n");
    }

    //设置 Traffic
    retVal = SetQHYCCDParam(camHandle, CONTROL_USBTRAFFIC, 1.0);    //设置 Traffic 为 1
```

```
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set traffic successfully.\n");
    }

    //获取 Traffic
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_USBTRAFFIC);
}
```

28.设置和获取白平衡的 RGB 分量

设置相机的 RGB 增益，通过此参数设置可以使图像更接近真实色彩。此设置只对彩色相机的连续模式生效，值越大，色彩占比越高，且和增益设置类似，可以随时设置。

设置 RGB 增益前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，并使用 GetQHYCCDParamMinMaxStep 函数确定参数设置范围，检查和参数设置范围只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep 和 SetQHYCCDParam，下为设置 RGB 增益的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
double min, max, step;

//查看是否支持红色白平衡
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_WBR);
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置范围
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_WBR, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get red gain range successfully.\n");
    }

    //设置红色白平衡参数
    retVal = SetQHYCCDParam(camHandle, CONTROL_WBR, 64.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set red gain successfully.\n");
    }

    //获取红色白平衡参数
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_WBR);
}

//查看是否支持绿色白平衡
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_WBG);
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置范围
```

```
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_WBG, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get green gain range successfully.\n");
    }

    //设置绿色白平衡参数
    retVal = SetQHYCCDParam(camHandle, CONTROL_WBG, 64.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set green gain successfully.\n").
    }

    //获取绿色白平衡参数
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_WBG);
}

//查看是否支持蓝色白平衡
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_WBB);
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置范围
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_WBB, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get blue gain range successfully.\n");
    }

    //设置蓝色白平衡参数
    retVal = SetQHYCCDParam(camHandle, CONTROL_WBB, 64.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set blue gain successfully.\n");
    }

    //获取蓝色白平衡参数
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_WBB);
}
```

29.设置和获取亮度

设置图像的亮度，这个设置是在 SDK 内部对图像数据进行处理，默认值为 0.0，设置值越大，图像越明亮，值越小，图像越暗，只对连续模式有效。

此参数无条件限制，可以随时设置。设置之前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，并使用 GetQHYCCDParamMinMaxStep 函数确定参数设置范围，检查和参数设置范围只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep 和 SetQHYCCDParam，下为设置亮度的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
```

```
double min, max, step;

//查看是否支持亮度
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_BRIGHTNESS);
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置范围和步长
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_BRIGHTNESS, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get brightness range successfully.\n");
    }

    //设置亮度
    retVal = SetQHYCCDParam(camHandle, CONTROL_BRIGHTNESS, 0.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set brightness successfully.\n);
    }

    //获取亮度值
    double value;
    value = GetQHYCCDParam(canHandle, CONTROL_BRIGHTNESS);
}
```

30.设置对比度

设置图像的对比度，这个设置是在 SDK 内部对图像数据进行处理，默认值为 0.0，设置值越大，图像越鲜亮，只对连续模式有效。

此参数无条件限制，可以随时设置。设置之前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，并使用 GetQHYCCDParamMinMaxStep 函数确定参数设置范围，检查和参数设置范围只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep 和 SetQHYCCDParam，下为设置对比度的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
double min, max, step;

//查看是否支持对比度
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_CONTRAST);
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置范围和步长
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_CONTRAST, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get contrast range successfully.\n");
    }

    //设置对比度
    retVal = SetQHYCCDParam(camHandle, CONTROL_CONTRAST, 0.0);
```

```
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set contrast Successfully.\n");
    }

    //获取对比度设置值
    double value;
    value = GetQHYCCDParam(canHandle, CONTROL_CONTRAST);
}
```

31.设置 GAMMA 值

设置图像的 Gamma 值，这个设置是在 SDK 内部对图像数据进行处理，默认值为 1，设置值越大，图像越亮，只对连续模式有效。

此参数无条件限制，可以随时设置。设置此参数之前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，并使用 GetQHYCCDParamMinMaxStep 函数确定参数设置范围，检查和参数设置范围只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep 和 SetQHYCCDParam，下为设置对比度的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
double min, max, step;

//查看是否支持 Gamma 设置
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_GAMMA);
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置范围和步长
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_GAMMA, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get gamma range successfully.\n");
    }

    //设置 Gamma
    retVal = SetQHYCCDParam(camHandle, CONTROL_GAMMA, 1.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set gamma successfully.\n");
    }

    //获取 Gamma 值
    double value;
    value = GetQHYCCDParam(camhandle, CONTROL_GAMMA);
}
```

32.设置图像传输速度

设置图像数据的传输速度，通过此设置可以缩短单帧模式下的读出时间或提高连续模式下的帧率。

设置此参数之前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，并使用

GetQHYCCDParamMinMaxStep 函数确定参数设置范围，检查和参数设置范围只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep 和 SetQHYCCDParam，下为设置传输速度的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
double min, max, step;

//查看是否支持速度设置
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_SPEED);
if(retVal == QHYCCD_SUCCESS)
{
    //获取参数设置范围和步长
    retVal = GetQHYCCDParamMinMaxStep(camHandle, CONTROL_SPEED, &min, &max, &step);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get speed range successfully.\n");
    }

    //设置速度
    retVal = SetQHYCCDParam(camHandle, CONTROL_SPEED, 0.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set speed successfully.\n");
    }

    //获取速度设置值
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_SPEED);
}
```

33.设置辉光抑制功能

设置相机辉光抑制功能的开启与关闭，参数为 1 时为开启，0 时为关闭，开启此功能可以减少图像中的辉光影响。

设置此参数之前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，检查只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable 和 SetQHYCCDParam，下为设置辉光抑制的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;

//查看是否支持辉光控制
retVal = IsQHYCCDControlAvailable(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
    //设辉光控制功能开启
    retVal = SetQHYCCDParam(camHandle, CONTROL_AMPV, 1.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set AMPV successfully.\n");
    }
```

```
    //设置辉光控制功能关闭
    retVal = SetQHYCCDParam(camHandle, CONTROL_AMPV, 0.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set AMPV successfully.\n");
    }

    //获取辉光控制功能设置值
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_AMPV);
}
```

34.设置 DDR

设置相机 DDR 功能开启或关闭，参数为 1 时为开启，0 时为关闭。DDR 开启时可以缓冲图像数据，避免数据丢包导致的图像丢失问题。单帧模式下默认设置 DDR 开启，切不可关闭，连续模式下可以根据需要选择开启或关闭。

设置此参数之前需要使用 IsQHYCCDControlAvailable 函数检查是否支持该设置，检查只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable 和 SetQHYCCDParam，下为设置辉光抑制的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;

//查看是否支持 DDR 功能
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_DDR);
if(reVal == QHYCCD_SUCCESS)
{
    //设置 DDR 开启
    retVal = SetQHYCCDParam(camHandle, CONTROL_DDR, 1.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set DDR successfully.\n");
    }

    //设置 DDR 关闭
    retVal = SetQHYCCDParam(camHandle, CONTROL_DDR, 0.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set DDR successfully.\n");
    }

    //获取 DDR 参数设置值
    double value;
    value = GetQHYCCDParam(camHandle, CONTROL_DDR);
}
```

35.设置行降噪

设置行降噪开启或关闭，功能开启时 SDK 内部会根据过扫区平均值进行计算，以此来降低水平随机条纹。目前只有 QHY5II-M 相机支持此功能。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，下为设置行降噪功能的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;

//查看是否支持行降噪功能
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_ROWNOISERE);
if(reVal == QHYCCD_SUCCESS)
{
	retVal = SetQHYCCDParam(camHandle, CONTROL_ROWNOISERE, 1.0);
	if(retVal == QHYCCD_SUCCESS)
	{
		printf("Set row noisere successfully.\n");
	}
}
```

36.设置 WDM 广播功能

开启或关闭 WDM 广播功能，通过此功能可以将视频图像发送到多个目标软件上。例如使用 SharpCap 连接具有 WDM 功能的相机，可以将 SharpCap 的显示视频图像发送到其他支持 WDM 的软件上进行显示，适用于视频在线广播类应用。

此功能只能在连续模式下使用，且暂时不支持 16 位图像数据发送，目前只支持发送八位的黑白和彩色图像数据。另外使用此功能需要安装 BroadCast WDM 驱动，此驱动可以通过官网下载界面的 AllInOne 安装包进行安装。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、SetQHYCCDParam，下为设置功能开启的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;

//查看是否支持广播功能
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_VCAM);
if(reVal == QHYCCD_SUCCESS)
{
	//设置广播功能开启
	retVal = SetQHYCCDParam(camHandle, CONTROL_VCAM, 1.0);
	if(retVal == QHYCCD_SUCCESS)
	{
		printf("Set row noisere successfully.\n");
	}

	//设置广播功能关闭
	retVal = SetQHYCCDParam(camHandle, CONTROL_VCAM, 0.0);
	if(retVal == QHYCCD_SUCCESS)
	{
		printf("Set row noisere successfully.\n");
	}

	//获取广播功能设置参数
	double value;
	value = GetQHYCCDParam(camHandle, CONTROL_VCAM);
}
```

37.设置和读取高低增益模式

设置高低增益，参数为 0 时是低增益模式，参数为 1 时是高增益模式，目前只有 QHY5III178 和 QHY178 使用此功能，其他相机已将高低增益设置合并至增益设置功能。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、SetQHYCCDParam，下为设置高低增益的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;

//检查是否支持高低增益切换
retVal = IsQHYCCDControlAvailable(camHandle, CAM_LIGHT_PERFORMANCE_MODE);
if(reVal == QHYCCD_SUCCESS)
{
    //设置低增益模式
    retVal = SetQHYCCDParam(camHandle, CAM_LIGHT_PERFORMANCE_MODE, 0.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set low gain mode successfully.\n");
    }

    //设置高增益模式
    retVal = SetQHYCCDParam(camHandle, CAM_LIGHT_PERFORMANCE_MODE, 1.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set high gain mode successfully.\n");
    }

    //获取高低增益模式设置值
    double value;
    value = GetQHYCCDParam(camHandle, CAM_LIGHT_PERFORMANCE_MODE);
}
```

38.设置 5II 系列相机导星模式开关

设置 5II 系列相机导星模式的开启和关闭，1 为开启，0 为关闭，默认为开启状态。

开启导星模式时将无法输出真正的十六位数据，输出的原始数据位数实际上还是八位的，相机对八位的原始数据低位补零转换成十六位数据，关闭后，相机可以产生 12 位原始数据，相机对十二位原始数据低位补零转换成十六位数据。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、SetQHYCCDParam，下为设置导星模式的示例代码：

```
retVal = IsQHYCCDControlAvailable(camHandle, CAM_QHY5II_GUIDE_MODE);
if(reVal == QHYCCD_SUCCESS)
{
    retVal = SetQHYCCDParam(camHandle, CAM_QHY5II_GUIDE_MODE, 1.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set QHY5II guide mode successfully.\n");
    }
}
```

39.单帧拍摄功能

控制相机的单帧拍摄功能，每次曝光只能获取一帧图像，若曝光未完成就开始调用 GetQHYCCDSingleFrame 函数获取图像，SDK 会在内部阻塞等待，直到曝光完成并读出数据。

连接相机并设置 BIN、分辨率、位数和彩色模式之后可以使用此功能，使用的函数有 ExpQHYCCDSingleFrame、GetQHYCCDSingleFrame 和 CancelQHYCCDExposingAndReadout，下为单帧模式示例代码：

```
int retVal = QHYCCD_ERROR;
int length = 0;
uint32_t w, h, bits, channels;
uint8_t *ImgData;

//获取图像数据内存长度并开辟存储空间
length = GetQHYCCDLength(camHandle);
ImgData = (uint8_t *)malloc(length);

//开始单帧模式曝光
retVal = ExpQHYCCDSingleFrame(camHandle);
if(retVal != QHYCCD_ERROR)
{
    printf("Start expose successfully.\n");

    //获取单帧模式图像数据
    retVal = GetQHYCCDSingleFrame(camHandle, &w, &h, &bits, &channels, ImgData);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Get single frame successfully.\n");
    }
}

//结束曝光及拍摄
retVal = CancelQHYCCDExposingAndReadout(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Stop single frame successfully.\n", pre)
}
```

拍摄完成后或有些情况下拍摄中途需要停止拍摄时，此时需要使用结束拍摄功能，示例代码如下：

```
retVal = CancelQHYCCDExposingAndReadout(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Stop single frame successfully.\n", pre)
}
```

停止拍摄之后需要调用 ExpQHYCCDSingleFrame 函数之后才能用 GetQHYCCDSingleFrame 函数继续获取图像数据。

40.连续拍摄功能

控制相机的连续拍摄功能，可以通过此功能从相机中获取视频数据流，根据视频流数据可以获取实时的连续图像。连续模式下需要持续调用 GetQHYCCDLiveFrame 函数以持续获取图像数据。

连接相机并设置 BIN、分辨率、位数和彩色模式之后可以使用此功能，使用的函数有 BeginQHYCCDLive、GetQHYCCDLiveFrame 和 StopQHYCCDLive，下为连续模式下拍摄三十帧图像的示例代码：

```
int retVal = QHYCCD_ERROR;
int length = 0, num = 0;
uint32_t w, h, bits, channels;
uint8_t *ImgData;

//获取图像数据内存长度并开辟存储空间
length = GetQHYCCDLength(camHandle);
ImgData = (uint8_t *)malloc(length);

//开始连续模式曝光
retVal = BeginQHYCCDLive(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
    while(num < 100)
    {
        //获取连续图像数据
        retVal = GetQHYCCDLiveFrame(camHandle, &w, &h, &bits, &channels, ImgData);
        if(retVal == QHYCCD_SUCCESS)
        {
            printf("Get live frame successfully.\n");
            num ++;
        }
    }
}

//结束连续模式曝光及拍摄
retVal = StopQHYCCDLive(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Stop live frame successfully.\n", pre)
}
```

拍摄完成后需要使用结束拍摄功能，示例代码如下：

```
retVal = StopQHYCCDLive(camHandle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Stop live frame successfully.\n")
}
```

停止拍摄之后需要调用 BeginQHYCCDLiveFrame 函数之后才能用 GetQHYCCDLiveFrame 函数获取图像数据，由于连续模式下使用循环调用的方式获取图像数据，因此先退出获取数据的循环之后再调用 StopQHYCCDLive 函数比较稳妥。

41.获取相机湿度传感器信息

获取相机湿度传感器的信息，此信息数据会持续变化，因此需要持续获取以得到实时信息。

获取此参数之前需要使用 IsQHYCCDControlAvailable 函数检查是否支持此功能，检查只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable 和 GetQHYCCDParam，下为获取湿度传感器信息的示例代码：

```
int retVal = QHYCCD_ERROR;
double hum;
retVal = IsQHYCCDControlAvailable(camHandle,CAM_HUMIDITY );
if(retVal == QHYCCD_SUCCESS)
{
    hum = GetQHYCCDParam(camHandle, CAM_HUMIDITY);
    printf("Camera humidity: %f\n", hum)
}
```

42.获取相机压力传感器信息

获取相机压力传感器的信息，此信息数据会持续变化，因此需要持续获取以得到实时信息。

获取此参数之前需要使用 IsQHYCCDControlAvailable 函数检查是否支持此功能，检查只需要确定一次。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable 和 GetQHYCCDParam，下为获取压力传感器信息的示例代码：

```
int retVal = QHYCCD_ERROR;
double pre;
retVal = IsQHYCCDControlAvailable(camHandle,CAM_PRESSURE );
if(retVal == QHYCCD_SUCCESS)
{
    pre = GetQHYCCDParam(camHandle, CAM_PRESSURE);
    printf("Camera pressure: %f\n", pre)
}
```

43.设置相机循环泵开关

设置相机循环泵的开启和关闭，1 为开启，0 为关闭。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、SetQHYCCDParam，下为设置循环泵开启的示例代码：

```
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_SensorChamberCycle_PUMP);
if(retVal == QHYCCD_SUCCESS)
{
    //开启循环泵
    retVal = SetQHYCCDParam(camHandle, CONTROL_SensorChamberCycle_PUMP, 1.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set pump on successfully.\n");
    }

    //关闭循环泵
    retVal = SetQHYCCDParam(camHandle, CONTROL_SensorChamberCycle_PUMP, 0.0);
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set pump off successfully.\n");
    }
}
```

44.相机温控功能

相机的制冷功能包括制冷控制，读取当前制冷功率和当前温度。

制冷控制分为手动控制和自动控制两种，手动控制是直接设置制冷器的制冷功率，使相机以某一固定

功率进行制冷。自动制冷则是只需要设置目标温度，相机会根据内部算法调节制冷功率，使相机温度在目标温度。

自动模式下，CMOS 和 CCD 相机的制冷控制略有差异，CMOS 相机会在相机驱动中自动调节制冷功率，可以只设置一次目标温度，也可以连续多次设置，不过 CCD 相机的驱动中没有这种功能，因此必须持续设置目标温度以实现自动调节功能。

相机的制冷功率和温度会持续变化，因此需要持续获取以得到实时信息。

另外，单帧模式下使用温控功能时需要在读取图像数据的时候暂时停止设置和获取相机温度，相机读取数据结束后再继续使用温控功能。

CMOS 相机的制冷范围一般为低于环境温度 35℃左右，当相机进行短曝光时会产生额外的热量，会使相机的制冷范围缩小。相对来说，CCD 相机的制冷性能要更优秀一些。另外，相机达到目标温度后不会立即稳定在目标温度，而是会波动一段时间，然后维持在目标温度。

设置制冷之前需要调用 IsQHYCCDControlAvailable 函数检查相机是否具有制冷功能。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParam 和 SetQHYCCDParam，下为相机制冷的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;
double pwm, temp, nowpwm, nowtemp;

//检查相机是否支持制冷功能
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_COOLER);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Camera has cooler function.\n");
}

//设置制冷功率，手动模式
pwm = 50.0;
retVal = SetQHYCCDParam(camHandle, CONTROL_MANULPWM, pwm / 100.0 * 255.0); //设置制冷功率为 50%
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set manual PWM successfully.\n");
}

//设置目标制冷温度，自动模式
temp = -10.0;
while(true)
{
    retVal = SetQHYCCDParam(camHandle, CONTROL_COOLER, temp);   //设置亮度为-10.0
    if(retVal == QHYCCD_SUCCESS)
    {
        printf("Set target temperature successfully.\n");
    }

    sleep(1000); //设置不必太频繁，每秒设置一次即可
}
```

```
//获取当前制冷器功率
while(true)
{
	nowpwm = GetQHYCCDParam(camHandle, CONTROL_CURPWM);
	printf("Now PWM : %f%%.\n", nowpwm / 255.0 * 100.0);

	sleep(1000); //每秒获取一次以得到实时数据
}

//获取当前温度
while(true)
{
	nowtemp = GetQHYCCDParam(camHandle, CONTROL_CURTEMP);
	printf("Now TEMP : %f.\n", nowtemp);

	sleep(1000);//每秒获取一次以得到实时数据
}

//关闭制冷
retVal = SetQHYCCDParam(camHandle, CONTROL_MANULPWM, 0.0);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Set cooler off successfully.\n");
}
```

45.滤镜轮控制功能

某些相机可以通过 4PIN 接口连接滤镜轮，并通过 SDK 直接控制滤镜轮，涉及到的功能有检查相机是否支持滤镜轮控制功能，检查滤镜轮是否已连接，获取滤镜轮孔数，设置目标孔位控制滤镜轮转动，获取滤镜轮当前位置信息。

滤镜轮孔位的设置范围为 0~cfwNum-1，cfwNum 为滤镜轮孔数。滤镜轮转动到目标位置时，会返回位置信息，因此可以持续获取滤镜轮位置信息，通过判断滤镜轮当前位置是否与目标位置相同来确认滤镜轮是否仍处于转动中。

设置滤镜轮位置时，SetQHYCCDParam 和 SendOrder2QHYCCDCFW 功能相同，区别在于参数设置方式，SetQHYCCDParam 设置的参数是 ASCII 码'0'、'1'、'2'...对应的值 48、4、50...，而 SendOrder2QHYCCDCFW 使用 char 类型的 ASCII 码作为参数进行设置，获取滤镜轮位置时，GetQHYCCDParam 和 GetQHYCCDCFWStatus 功能相同，区别为 GetQHYCCDParam 获取到的滤镜轮位置为 ASCII 码对应的值，如 48、49、50 等，GetQHYCCDCFWStatus 获取到的值为 char 类型的 ASCII 码。

另外，获取滤镜轮位置时，若使用 C/C++进行开发，使用哪个函数设置或获取滤镜轮位置都可以，若是使用其他语言建议使用 SetQHYCCDParam 和 GetQHYCCDParam 函数，否则可能会无法正确设置目标位置或获取位置信息。

通过 InitQHYCCD 函数初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、IsQHYCCDCFWPlugged、SetQHYCCDParam（SendOrder2QHYCCDCFW）、GetQHYCCDParam（GetQHYCCDCFWStatus）函数，SetQHYCCDParam 与 SendOrder2QHYCCDCFW 函数功能相同，GetQHYCCDParam 与 GetQHYCCDCFWStatus 函数功能相同。下为滤镜轮控制的示例代码：

```
uint32_t retVal = QHYCCD_ERROR;

//检查是否支持滤镜轮控制
retVal = IsQHYCCDControlAvailable(camHandle, CONTROL_CFWPORT);
if(retVal == QHYCCD_SUCCESS)
{
	printf("This camera can control CFW.\n");
}

//获取滤镜轮孔数
int posNum;
posNum = GetQHYCCDParam(camHandle, CONTROL_CFWSLOTSNUM);
printf("This CFW has %s slots.\n");

//设置滤镜轮目标位置为第三孔，方式一
retVal = SetQHYCCDParam(camHandle, CONTROL_CFWPORT, 50.0);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Setup target position successfully.\n");
}
//设置滤镜轮目标位置为第三孔，方式二
retVal = SendOrder2QHYCCDCFW(camHandle, '2', 1);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Setup target position successfully.\n");
}

//获取滤镜轮位置信息以判断滤镜轮运行状态，方式一
double nowPos;
while(nowPos != targetPos)
{
	nowPos = GetQHYCCDParam(camHandle, CONTROL_CFWPORT);
	printf("Now CFW position is %c.\n", (int)nowPos);
}
//获取滤镜轮位置信息以判断滤镜轮运行状态，方式二
char nowPos;
while(nowPos != targetPos)
{
	retVal = GetQHYCCDCFWStatus(camHandle, &nowPos);
	if(retVal == QHYCCD_SUCCESS)
	{
		printf("Now CFW position is %c.\n", nowPos);
	}
}
```

46.设置 GPS

通过 GPS 功能可以对相机曝光时间参数进行测量，以获取帧经纬度信息、曝光开始和结束的精确时间以及实现相机的同步操作。

GPS 功能开启之后需要等待 GPS 信号锁定，锁定之后 GPS 模块会将 GPS 信息传输给相机，另外受信号

强度影响，GPS 信号锁定所需时间并不固定。

目前相机获取 GPS 信息方式分为两种，第一种是带有内置 GPS 模块的相机，这类相机可以设置主从模式，以及进行 PPS 校准和 LED 灯校准操作，现在内置 GPS 模块的相机有 QHY174GPS、QHY550、QHY1253、QHY990、QHY991，另外一种是使用外置 GPS 模块的相机，这类相机只能设置 GPS 开启关闭，无法设置主从模式以及 PPS、LED 灯校准，并且由于测量波形限制，相机只能输出准确记录开始时间，其结束时间和开始时间会固定相差 400ns，现在外置 GPS 模块的相机有 QHY600PRO、QHY268PRO、QHY411、QHY461、QHY4040、QHY4040PRO、QHY1920、QHY342PRO、QHY42PRO、QHY163、QHY6060PRO。

内置 GPS 模块的相机可以设置两种工作模式，分别是主模式和从模式。主模式指的就是一般的工作模式，可以支持 ROI。从模式下相机由时间触发，根据设置的开始时间、曝光时间和曝光间隔等参数执行拍摄任务，从模式不支持 ROI。

内置 GPS 模块的相机需要在 GPS 信号锁定后对 VCOX 和 LED 进行校准。VCOX 校准用于消除温度漂移问题对 PPS 计数器的影响，LED 校准则是为了得到更为精准的曝光开始和结束时间。

VCOX 校准的具体操作为调节 VCOX 频率，使 PPS 计数器值接近 1000000，当 PPS 信号丢失时，PPS 计数器值将变成 10000500，因此不要让 PPS 计数器值超过 10000500，以避免混乱。

LED 校准是通过改变 Position A 和 Position B 来获取曝光开始和结束的精确时间。具体操作为观察 CMOS 旁边的 LED 灯，当增加 Position A，发现 LED 脉冲从不可见到可见时，为快门启动时间，这个位置是起始位置。当增加 Pposition B，发现 LED 脉冲从可见到不可见，这个位置是结束的位置。每次曝光时间和 USBTRAFFIC 改变时都需要重新校准 Position A 和 Position B，下为提前测试好的不同曝光时间下 Position A 和 Position B 的设置值。

8BIT USBTRAFFIC=0			16BIT USBTRAFFIC=0		
EXPOSURE（ms）	START POS（B）	END POS（A）	EXPOSURE（ms）	START POS（B）	END POS（A）
1.0	486080	2850	1.0	902840	4190
2.0	411270	2850	2.0	602930	4190
5.0	185930	2850	5.0	185930	4190
7.0	36290	2850	10.0	228050	4190
8.0	6280	2850	20.0	10930	4190
10.0	6280	2850	30.0	10930	4190
20.0	6280	2850	40.0	10930	4190
40.0	6280	2850	100.0	10930	4190
100.0	6280	2850	200.0	10930	4190
200.0	6280	2850	500.0	10930	4190
500.0	6280	2850			

单帧模式和连续模式都可以产生时间信息，接收到的 GPS 数据会存储在图像数据的前四十四个数据中，QHY174GPS、QHY550、QHY1253、QHY990、QHY991 这种带有内置 GPS 的相机会在四十四个 GPS 数据之后额外存储一些原始 GPS 信息数据。下为各位数据的说明及计算方式：

GPS 数据结构：

序列号			
0	Sequence Number MSB	1	Sequence Number
2	Sequence Number	3	Sequence Number LSB
4	temporary Sequence Number (Normally no use)		
图像宽度			
5	Image Width MSB	6	Image Width LSB
图像高度			
7	Image Height MSB	8	Image Height LSB
纬度			
9	latitude MSB	10	latitude
11	latitude	12	latitude LSB
经度			
13	longitude MSB	14	longitude
15	longitude	16	longitude LSB
快门开始时间(JS)			
17	Start_Flag	18	Start Second MSB
19	Start Second	20	Start Second
21	Start Second LSB	22	Start micro second MSB
23	Start micro second	24	Start micro second LSB
快门结束时间(JS)			
25	End flag	26	End Second MSB
27	End Second	28	End Second
29	End Second LSB	30	End micro second MSB
31	End micro second	32	End micro second LSB
GPS 状态			
33	now flag		
当前时间（CMOS 传感器的垂直同步时间，不是快门开始或关闭的精确时间）			
34	now second MSB	35	now second
36	now second	37	now second LSB
38	now micro second MSB	39	now micro second
40	now micro second LSB		
PPS 计数器值			
41	count of PPS MSB	42	count of PPS
43	count of PPS LSB		

数据转换方法：

```
//帧序列号
int seqNumber, tempNumber;
seqNumber = 256*256*256*imageHead[0]+256*256*imageHead[1]+256*imageHead[2]+imageHead[3];
tempNumber = imageHead[4];

//图像宽度
int width;
width = 256*imageHead[5]+imageHead[6];

//图像高度
```

```
int height;
height = 256*imageHead[7]+imageHead[8];

//纬度
int temp, deg, min, south;
double fractMin, latitude;
temp = 256*256*256*imageHead[9]+256*256*imageHead[10]+256*imageHead[11]+imageHead[12];
south = temp > 1000000000;
deg = (temp % 1000000000) / 10000000;
min = (temp % 10000000) / 100000;
fractMin = (temp % 100000) / 100000.0;
latitude = (deg + (min + fractMin) / 60.0) * (south==0 ? 1 : -1);

//经度
int temp, deg, min, west;
double fractMin, longitude;
temp = 256*256*256*imageHead[13]+256*256*imageHead[14]+256*imageHead[15]+imageHead[16];
west = temp > 1000000000;
deg = (temp % 1000000000) / 1000000;
min = (temp % 1000000) / 10000;
fractMin = (temp % 10000) / 10000.0;
longitude = (deg + (min + fractMin) / 60.0) * (west==0 ? 1 : -1);

//快门开始时间
int start_flag, start_sec, start_us;
start_flag = imageHead[17];
start_sec = 256*256*256*imageHead[18]+256*256*imageHead[19]+256*imageHead[20]+imageHead[21];
start_us = 256*256*imageHead[22]+256*imageHead[23]+imageHead[24];

//快门结束时间
int end_flag, end_sec, end_us;
end_flag = imageHead[25];
end_sec = 256*256*256*imageHead[26]+256*256*imageHead[27]+256*imageHead[28]+imageHead[29];
end_us = 256*256*imageHead[30]+256*imageHead[31]+imageHead[32];

//GPS 状态
int now_flag;
now_flag = (imageHead[33] / 16 ) %4;

//当前时间
int now_sec, now_us;
now_sec = 256*256*256*imageHead[34]+256*256*imageHead[35]+256*imageHead[36]+imageHead[37];
now_us = 256*256*imageHead[38]+256*imageHead[39]+imageHead[40];

//PPS 计数值
int pps;
pps = 256*256*imageHead[41]+256*imageHead[42]+imageHead[43];

//曝光时间
double expose
exposure = (end_sec - start_sec) * 1000 * 1000 + (end_us - start_us);
```

```
//原始 GPS 数据
int i;
int rawHeadPosition=0;
for (i = 34; i < 1024; i++)
{
	if(ImgDataGPS[i]==0x11)
	{
		if(ImgDataGPS[i+1]==0x22 && ImgDataGPS[i+2]==0x33 && ImgDataGPS[i+3]==0x66)
		{
			rawHeadPosition=i;
		}
	}
}
int rawTailPosition=0;
for (i = 34; i < 1024; i++)
{
	if(ImgDataGPS[i]==0xee)
	{
		if(ImgDataGPS[i+1]==0x33 && ImgDataGPS[i+2]==0xcc && ImgDataGPS[i+3]==0x44)
		{
			rawTailPosition=i;
		}
	}
}

int GPS_RAW_LENGTH;
GPS_RAW_LENGTH = ImgDataGPS[rawHeadPosition+4]*256*256*256 +
                 ImgDataGPS[rawHeadPosition+5]*256*256 +
                 ImgDataGPS[rawHeadPosition+6]*256 +
                 ImgDataGPS[rawHeadPosition+7];

char rawstr[1024];
if(rawHeadPosition+8+GPS_RAW_LENGTH<1024)
{
	//get the raw data (raw head position +8)
	for (int j = 0; j < GPS_RAW_LENGTH; j++)
	{
	   rawstr[j] = ImgDataGPS[j+rawHeadPosition+8];
	}
}
string gpsRawData(rawstr);
```

时间格式转换：

start_sec、end_sec、now_sec 为从 1995 年 10 月 10 号 0 点 0 分 0 秒到现在为止的总秒数，可将将其转换成年月日时分秒的格式，下面为转换代码：

```
typedef struct
{
uint16_t year;
```

```
uint16_t month;
uint16_t date;
uint16_t hour;
uint16_t min;
uint16_t sec;
uint16_t week;
}drive_time,*pdrive_time;

drive_time UTC;

drive_time struct_time =   //初始化时间
{
.year = 1995,
.month = 10,
.date = 10,
.hour = 0,
.min = 0,
.sec = 0,
};

bool isLeapYear( int year )
{
	if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0))
		return true;

	return false;
}

//获取世界标准时间，转换成北京时间需要加上 8 小时
int get_UTC(unsigned long second, pdrive_time UTC)
{
	const char Leap_Year_day[2][12] =
{ {31,28,31,30,31,30,31,31,30,31,30,31},{31,29,31,30,31,30,31,31,30,31,30,31} };
	int Leap_Year = 0;
	int month_day = 0;

	Leap_Year = isLeapYear(struct_time.year);
	month_day = Leap_Year_day[Leap_Year][struct_time.month-1];

	UTC->year = struct_time.year;
	UTC->month = struct_time.month;
	UTC->date = struct_time.date;
	UTC->hour = struct_time.hour +(second / 3600 % 24);
	UTC->min = struct_time.min+ (second / 60 % 60);
	UTC->sec = struct_time.sec +(second % 60);

	uint16_t count_days = second / 86400;

	if(UTC->sec >=60)
	{
		UTC->sec = UTC->sec%60;
		(UTC->min) ++;
	}
```

```
    if(UTC->min >=60)
    {
        UTC->min = UTC->min%60;
        (UTC->hour) ++;
    }
    if(UTC->hour >=24)
    {
        UTC->hour = UTC->hour%24;
        (count_days) ++;
    }

    for(int i = 0 ; i < count_days; i++ )
    {
        Leap_Year = isLeapYear(UTC->year);
        month_day = Leap_Year_day[Leap_Year][(UTC->month)-1];

        (UTC->date) ++;
        if((UTC->date) > month_day)
        {
            (UTC->date) = 1;
            (UTC->month) ++;
            if((UTC->month) > 12)
            {
                (UTC->month) = 1;
                (UTC->year) ++;
                if( ( (UTC->year) - (struct_time.year) ) >100)
                return -1;
            }
        }
    }

    return    0 ;
}

get_UTC(start_sec, &UTC);
printf("%d %d %d %d %d %d\n", UTC.year, UTC.month, UTC.date, UTC.hour, UTC.min, UTC.sec);
```

通过 InitQHYCCD 函数初始化相机之后可以对此功能进行设置，但是只有获取图像数据之后才能得到 GPS 信息，使用的函数有 SetQHYCCDParam、SetQHYCCDGPSVCOXFreq、SetQHYCCDGPSLedCalMode、SetQHYCCDGPSPOSA、SetQHYCCDGPSPOSB、SetQHYCCDGPSMasterSlave、SetQHYCCDGPSSlaveModeParameter，下为连续模式下拍摄 200 帧图像，并计算 GPS 数据的示例代码：

```
int retVal = QHYCCD_ERROR;
unsigned char gps[44] = { 0 };
int length = 0;
int imagew, imageh, bits, channels;
int seqNumber, tempNumber, width, height;
```

```
int status, pps;
int temp, deg, min, south, west;
double fractMin, latitude, longitude;
int start_sec, start_us, end_sec, end_us, now_sec, now_us;
double exposure;

//检查是否支持 GPS 功能
retVal = IsQHYCCDControlAvailable(camHandle, CAM_GPS);
if(retVal == QHYCCD_SUCCESS)
{
    printf("The camera can use GPS function.\n");
}


//打开 GPS 功能
retVal = SetQHYCCDParam(camHandle, CAM_GPS, 1.0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Turn GPS on successfully.\n");
}


//此设置需要根据实际情况进行调节，使 PPS 计数值接近 1000000 即可，仅 QHY174-GPS 需要
retVal = SetQHYCCDGPSVCOXFreq(camHandle, 2);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Set GPS VCOX frequency successfully.\n");
}


//设置校准脉冲的位置 A 和位置 B，需要根据实际情况设置，仅 QHY174-GPS 需要
SetQHYCCDGPSPOSA(camHandle, 0, pos, 40);
SetQHYCCDGPSPOSB(camHandle, 0, pos, 40);

length = GetQHYCCDMemLength(camHandle);
ImgData = (uint8_t *)malloc(length);

int num = 0;
//开始连续模式拍摄
retVal = BeginQHYCCDLive(camHandle);
while(num < 200)
{
    retVal = GetQHYCCDLiveFrame(camHandle, &imagw, &imageh, &bits, &channels, ImgData);
    if(retVal == QHYCCD_SUCCESS)
    {
```

```
            retVal = QHYCCD_ERROR;
            num ++;

            memcpy(gps, ImgData, 44);
            //GPS 状态
            now_flag = (gps[33] / 16 ) %4;
            //PPS 计数值
            pps = 256*256*gps[41] + 256*gps[42] + gps[43];
            //帧序号
            seqNumber = 256*256*256*gps[0] + 256*256*gps[1] + 256*gps[2] + gps[3];
            //图像宽度
            width = 256*gps[5] + gps[6];
            //图像高度
            height = 256*gps[7] + gps[8];
            //纬度
            temp = 256*256*256*gps[9] + 256*256*gps[10] + 256*gps[11] + gps[12];
            south = temp > 1000000000;
            deg = (temp % 1000000000) / 10000000;
            min = (temp % 10000000) / 100000;
            fractMin = (temp % 100000) / 100000.0;
            latitude = (deg + (min + fractMin) / 60.0) * (south==0 ? 1 : -1);
            //经度
            temp = 256*256*256*gps[13] + 256*256*gps[14] + 256*gps[15] + gps[16];
            west = temp > 1000000000;
            deg = (temp % 1000000000) / 1000000;
            min = (temp % 1000000) / 10000;
            fractMin = (temp % 10000) / 10000.0;
            longitude = (deg + (min + fractMin) / 60.0) * (west==0 ? 1 : -1);
            //快门开始时间
            start_sec = 256*256*256*gps[18] + 256*256*gps[19] + 256*gps[20] + gps[21];
            start_us = 256*256*gps[22] + 256*gps[23] + gps[24];
            //快门结束时间
            end_sec = 256*256*256*gps[26] + 256*256*gps[27] + 256*gps[28] + gps[29];
            end_us = 256*256*gps[30] + 256*gps[31] + gps[32];
            //当前时间
            now_sec = 256*256*256*gps[34] + 256*256*gps[35] + 256*gps[36] + gps[37];
            now_us = 256*256*gps[38] + 256*gps[39] + gps[40];
            //曝光时间
            exposure = (end_sec - start_sec) * 1000 * 1000 + (end_us - start_us);
        }
}


//停止连续模式拍摄

retVal = StopQHYCCDLive(camHamdle);
if(retVal == QHYCCD_SUCCESS)
```

```
{
	printf("Stop live capture successfully.\n");
}


//关闭 LED 和 GPS 功能
retVal = SetQHYCCDGPSLedCalMode(camHandle, 0);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Turn GPS LED off successfully.\n");
}
retVal = SetQHYCCDParam(camHandle, CAM_GPS, 0);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Turn GPS off successfully.\n");
}
```

单帧模式下使用 GPS 功能的方式和连续模式类似，也是使用图像数据的前 44 个数据进行计算，不过区别在于，单帧模式每次只拍摄一帧图像，所以如果想获取实时的 GPS 信息需要持续拍摄，下为拍摄一帧单帧图像并计算 GPS 信息的示例代码：

```
int retVal = QHYCCD_ERROR;
unsigned char gps[44] = { 0 };
int length = 0;
int imagew, imageh, bits, channels;
int seqNumber, tempNumber, width, height;
int status, pps;
int temp, deg, min, south, west;
double fractMin, latitude, longitude;
int start_sec, start_us, end_sec, end_us, now_sec, now_us;
double exposure;

//检查是否支持 GPS 功能
retVal = IsQHYCCDControlAvailable(camHandle, CAM_GPS);
if(retVal == QHYCCD_SUCCESS)
{
	printf("The camera can use GPS function.\n");
}

//打开 GPS 功能
retVal = SetQHYCCDParam(camHandle, CAM_GPS, 1.0);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Turn GPS on successfully.\n");
}
```

```
//此设置需要根据实际情况进行调节，使 PPS 计数值接近 1000000 时即可，仅 QHY174-GPS 需要
retVal = SetQHYCCDGPSVCOXFreq(camHandle, 2);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Set GPS VCOX frequency successfully.\n");
}

//设置校准脉冲的位置 A 和位置 B，需要根据实际情况设置，仅 QHY174-GPS 需要
SetQHYCCDGPSPOSA(camHandle, 0, pos, 40);
SetQHYCCDGPSPOSB(camHandle, 0, pos, 40);

length = GetQHYCCDMemLength(camHandle);
ImgData = (uint8_t *)malloc(length);

int num = 0;
//开始连续模式拍摄
retVal = ExpQHYCCDSingleFrame(camHandle);
retVal = GetQHYCCDSingleFrame(camHandle, &imagw, &imageh, &bits, &channels, ImgData);
if(retVal == QHYCCD_SUCCESS)
{
	retVal = QHYCCD_ERROR;
	num ++;

	memcpy(gps, ImgData, 44);
	//GPS 状态
	now_flag = (gps[33] / 16 ) %4;
	//PPS 计数值
	pps = 256*256*gps[41] + 256*gps[42] + gps[43];
	//帧序号
	seqNumber = 256*256*256*gps[0] + 256*256*gps[1] + 256*gps[2] + gps[3];
	//图像宽度
	width = 256*gps[5] + gps[6];
	//图像高度
	height = 256*gps[7] + gps[8];
	//纬度
	temp = 256*256*256*gps[9] + 256*256*gps[10] + 256*gps[11] + gps[12];
	south = temp > 1000000000;
	deg = (temp % 1000000000) / 10000000;
	min = (temp % 10000000) / 100000;
	fractMin = (temp % 100000) / 100000.0;
	latitude = (deg + (min + fractMin) / 60.0) * (south==0 ? 1 : -1);
	//经度
	temp = 256*256*256*gps[13] + 256*256*gps[14] + 256*gps[15] + gps[16];
	west = temp > 1000000000;
	deg = (temp % 1000000000) / 1000000;
	min = (temp % 1000000) / 10000;
	fractMin = (temp % 10000) / 10000.0;
	longitude = (deg + (min + fractMin) / 60.0) * (west==0 ? 1 : -1);
	//快门开始时间
```

```
    start_sec = 256*256*256*gps[18] + 256*256*gps[19] + 256*gps[20] + gps[21];
    start_us = 256*256*gps[22] + 256*gps[23] + gps[24];
    //快门结束时间
    end_sec = 256*256*256*gps[26] + 256*256*gps[27] + 256*gps[28] + gps[29];
    end_us = 256*256*gps[30] + 256*gps[31] + gps[32];
    //当前时间
    now_sec = 256*256*256*gps[34] + 256*256*gps[35] + 256*gps[36] + gps[37];
    now_us = 256*256*gps[38] + 256*gps[39] + gps[40];
    //曝光时间
    exposure = (end_sec - start_sec) * 1000 * 1000 + (end_us - start_us);
}

//停止连续模式拍摄
retVal = CancelQHYCCDExposingAndReadout(camHamdle);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Stop live capture successfully.\n");
}

//关闭 LED 和 GPS 功能
retVal = SetQHYCCDGPSLedCalMode(camHandle, 0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Turn GPS LED off successfully.\n");
}
retVal = SetQHYCCDParam(camHandle, CAM_GPS, 0);
if(retVal == QHYCCD_SUCCESS)
{
    printf("Turn GPS off successfully.\n");
}
```

内置 GPS 模块的相机可以使用从模式设置定时拍摄，例如设置相机在 2023-12-4 15:51:00 UTC 拍摄，设置之后相机会处于静止状态，不再输出图像数据，当到达预设时间之后相机自动拍摄并输出图像数据，拍摄完成后相机会回到静止状态，下为设置代码：

```
int Date2Jd(int y, int m, int d) {
    int ky = y;
    int km = m;
    int B = -2;

    if (km <= 2) {
        ky = ky - 1;
        km = km + 12;
    }
    if (ky > 1582 || (ky == 1582 && (km > 10 || (km == 10 && d >= 15)))) {
        B = ky / 400 - ky / 100;
    }
```

```
    double JD = static_cast<int>(365.25 * ky) + static_cast<int>(30.6001 * (km + 1)) + B + 1720996.5 + d;

    return static_cast<int>(JD);
}

double Date2Js1995(int y, int M, int d, int h, int m, int s)
{
    double Jd;
    double Jds;
    double Jd1995;
    double Js1995;

    Jd = Date2Jd(y, M, d);
    Jds = Jd + static_cast<double>(h) / 24 + static_cast<double>(m) / 60 / 24 + static_cast<double>(s) / 60 / 60 /
24;
    Jd1995 = Jd - 2450000; // Zero time is 19951009
    Js1995 = Jd1995 * 24 * 3600 + h * 3600 + m * 60 + s;

    return Js1995;
}

SetQHYCCDParam(camhandle, CAM_GPS, 1.0);
SetQHYCCDGPSMasterSlave(camhandle, 1);
double target = Date2Js1995(2023, 12, 4, 15, 51, 0);//必须设置 UTC 时间
uint32_t exptime = 200000;//200ms
SetQHYCCDGPSSlaveModeParameter(camhandle, target, 0, 0, 0, exptime);
```

47.设置 AntiRBI 模式

设置相机的 AntiRBI 模式，此模式可以消除残影的影响。开启此模式后，相机将自动以一帧短曝光图像，一帧正常曝光图像间隔的方式开始拍摄，从图像上看，相机会以一亮一暗的方式向外输出图像，此功能只能在连续模式下使用。

使用的函数为 SetQHYCCDEnableLiveModeAntiRBI，下为示例代码：

```
int ret = QHYCCD_ERROR;
int num = 0, length = 0;
int imagew, imageh, bits, channels;

//开启 AntiRBI 模式
ret = SetQHYCCDEnableLiveModeAntiRBI(camHandle, 1);
if(ret == QHYCCD_SUCCESS)
{
    printf("Enable live mode AntiRBI successfully.\n");
}

//获取图像数据内存长度并开辟内存空间
length = GetQHYCCDMemLength(camHandle);
ImgData = (uint8_t *)malloc(length);
```

```
//开始连续模式曝光
ret = BeginQHYCCDLive(camHandle);
if(ret == QHYCCD_SUCCESS)
{
	printf("Begin live successfully.\n");
}

while(num < 100)
{
	//获取连续图像数据
	ret = GetQHYCCDLiveFrame(camHandle, &imagew, &imageh, &bits, &channels, ImgData);
	if(ret == QHYCCD_SUCCESS)
	{
		printf("Get live frame successfully.\n");
		num ++;
	}
}
```

48.设置 Burst 模式

Burst 模式是连续模式的一个子模式，在此模式下相机不会连续地输出图像数据，而是会等待开始拍摄的信号，当接收到信号之后，相机会连续输出几帧图像，输出图像的帧数可以设置。

另外需要注意的是，

1.光纤模式下使用 Burst 模式时，第一次 Burst 拍摄会少一张，例如设置 start end 为 1 6，输出 2 3 4 5 为正常，而实际上第一次 Burst 拍摄时只会输出 3 4 5，2 会无法收到，第二次及之后的拍摄可以正常获取 Burst 图像 2 3 4 5。这个问题将在后面进行修复。

2.QHY2020,QHY4040 发现曝光时间短的时候出来的帧序号是 [start+1,end-1] 但是长曝光下出来的是 [start+2,end]

3.相机刚连接时如果设置的 end 值比较大，相机进入 burst 模式后会直接出图，因此需要设置相机进入 IDLE 状态后再设置 start end 及相关 burst 操作

连续模式下可以使用此功能，使用的函数有 EnableQHYCCDBurstMode、EnableQHYCCDBurstCountFun、ResetQHYCCDFrameCounter、SetQHYCCDBurstModeStartEnd、SetQHYCCDBurstIDLE、ReleaseQHYCCDBurstIDLE、SetQHYCCDBurstModePatchNumber，下为 Burst 模式的示例代码：

```
int ret = QHYCCD_ERROR;
int num = 0, length = 0;
```

```
int start, end;
int imagew, imageh, bits, channels;

//获取图像数据内存长度并开辟内存空间
length = GetQHYCCDMemLength(camHandle);
ImgData = (uint8_t *)malloc(length);

//开启 burst 模式
ret = EnableQHYCCDBurstMode(camHandle, true);
if(ret == QHYCCD_SUCCESS)
{
	printf("Enable burst mode successfully.\n");
}

//设置开始帧和结束帧的位置，相机将输出中间帧
start = 1;
end = 3;
ret = SetQHYCCDBurstModeStartEnd(camHandle, start, end);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set burst mode start end successfully.\n");
}

//设置补包数据量，避免图像数据无法输出
ret = SetQHYCCDBurstModePatchNumber(camhandle, 32001);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set burst mode patch number successfully.\n");
}

//开始连续模式曝光
ret = BeginQHYCCDLive(camHandle);
if(ret == QHYCCD_SUCCESS)
{
	printf("Begin live successfully.\n");
}

//设置相机输出图像数据，此设置需要 SetQHYCCDBurstIDLE 和 ReleaseQHYCCDBurstIDLE 一起使用
ret = SetQHYCCDBurstIDLE(camHandle);
if(ret == QHYCCD_SUCCESS)
{
	printf("Set burst mode IDLE successfully.\n");

	ret = ReleaseQHYCCDBurstIDLE(camHandle);
	if(ret == QHYCCD_SUCCESS)
	{
		printf("Release burst mode IDLE successfully.\n");
	}
}

//获取连续图像数据
```

```
while(true)
{
	retVal = GetQHYCCDLiveFrame(camHandle, &imagew, &imageh, &bits, &channels, ImgData);
	if(retVal == QHYCCD_SUCCESS)
	{
		printf("Get live frame successfully.\n");
		num ++;
	}
}

//注：拍摄 burst 模式图像之后不需要结束连续模式，所有拍摄完成后再结束连续模式拍摄，不结束连续模式也可以重新设置 start、 end
```

49.设置外触发模式

设置相机的触发功能，触发功能分为两种，分别是触发输入和触发输出。

触发输入功能为，从外部发送触发信号以控制相机拍摄，触发模式使能时将无法通过调用原有的曝光函数来控制相机完成拍摄任务，此时需要从外部输入触发信号给相机，相机收到触发信号中后就会开始曝光，曝光完成后调用获取图像数据函数即可获取图像数据，触发信号可以由软件产生，也可以由硬件产生。

触发输出功能为，相机拍摄图像时会输出与曝光相关的脉冲，可以根据此脉冲获得相关的曝光信息。

单帧和连续模式都可以使用触发功能，不同相机的触发功能有所差异，可以同时使用触发输入和触发输出功能，也可以只使用其中一个。

连接相机后可以使用此功能，但必须在设置 BIN、分辨率、位数和彩色模式之后才能进行拍摄任务，使用的函数有 IsQHYCCDControlAvailable、 GetQHYCCDParamMinMaxStep、 SetQHYCCDTrigerFunction、 SetQHYCCDTrigerMode，下为单帧模式下使用此功能的示例代码：

```
int retVal = QHYCCD_ERROR;
double min, max, step;

//检查相机是否支持触发功能
retVal = IsQHYCCDControlAvailable(camHandle, CAM_TRIGER_INTERFACE);
iif(retVal == QHYCCD_SUCCESS)
{
	printf("This camera can use trigger function.\n");
}

//获取触发接口的数量和名称
uint32_t num = 0;
retVal = GetQHYCCDTrigerInterfaceNumber(camhandle, &num);
if(num > 1)
```

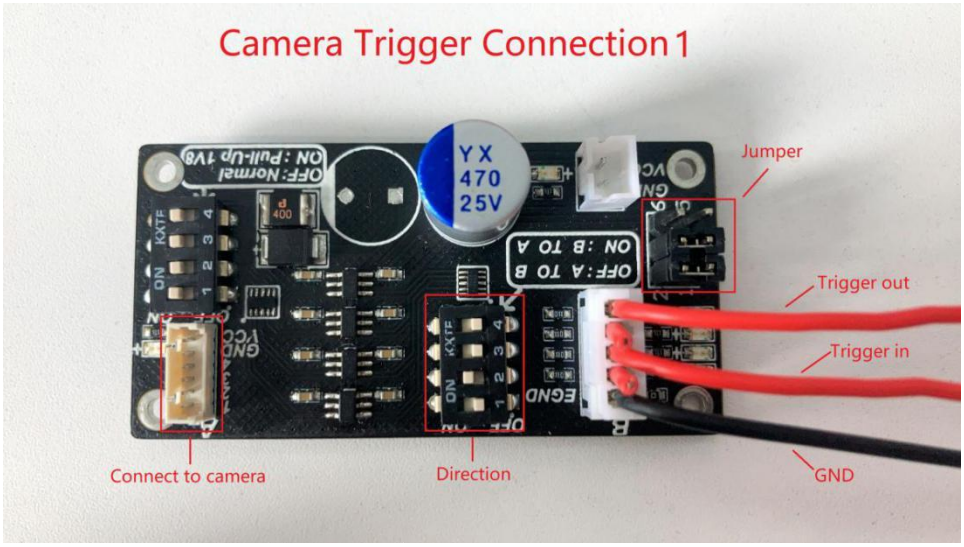
```
{
	char name[40] = { 0 };
	for(int i = 0; i < num; i ++)
	{
		retVal = GetQHYCCDTrigerInterName(camhandle, i, name);
		if(retVal == QHYCCD_SUCCESS)
		{
			printf("Get triger interface successfully.\n");
		}
	}
}

//设置触发接口
retVal = SetQHYCCDTrigerInterface(camhandle, 0);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Set interface successfully.\n");
}

//使能触发功能
retVal = SetQHYCCDTrigerFunction(camHandle, true);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Enable trigger mode successfully.\n");
}

//使能触发输出功能
retVal = EnableQHYCCDTrigerOut(camhandle);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Enable trigger out successfully.\n");
}

//开始单帧拍摄
retVal = ExpQHYCCDSingleFrame(camHandle);
if(retVal != QHYCCD_ERROR)
{
	printf("Start single expose successfully.\n");
}

//阻塞获取单帧数据
retVal = GetQHYCCDSingleFrame(camHandle, &imagew, &imageh, &bpp, &channels, ImgData);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Get single frame successfully.\n");
}

//结束单帧拍摄
retVal = CancelQHYCCDExposingAndReadout(camhandle);
if(retVal == QHYCCD_SUCCESS)
{
	printf("Stop single capture successfully.\n");
```

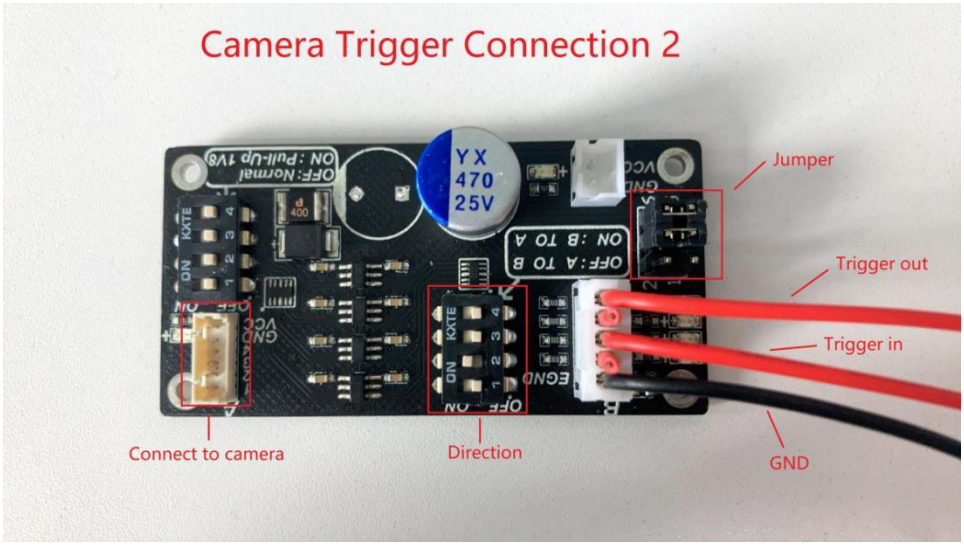
```
}
```

触发控制硬件说明：

目前相机的触发接口有两种，一种是 GPIO 接口，另一种是 SMA 接口。

若使用 GPIO 接口，我们建议使用配套的电平转换板进行控制。可以通过 GPIO 接口控制触发的相机有 QHY4040、QHY4040PRO、QHY600、QHY268、QHY461、QHY411，这些相机均为下降沿触发，但是触发电平标准不同，QHY4040 是 2.5V 触发，QHY4040PRO、QHY600、QHY268、QHY461、QHY411 是 1.8V 触发，以带有 6PIN GPIO 接口的相机为例，下两图为电平转换板的连接及拨动开关方向示意图：





这两种连接方式只有跳线方式不同，Connection1 的 Trigger Out 电平固定为 5V，Connection2 的 Trigger Out 电平为 1.8V 或 2.5V，QHY4040PRO、QHY600、QHY268、QHY461、QHY411 为 1.8V，QHY4040 为 2.5V。另外需要注意的是，QHY4040 及 QHY4040PRO 由于硬件原因，只能同时使用 Trigger In 或 Trigger Out 中的一个功能。

电平转换板说明：

http://note.youdao.com/noteshare?id=4b265780689cadb651452299be5b19dd&sub=6D04551AD06E41548FE9D341BC9DC48E

目前使用 SMA 触发接口的相机有 QHY4040、QHY411ERIS、QHY461、QHY550、QHY990、QHY991、QHY21、QHY22、QHY23、A 系列相机，所有使用 SMA 接口触发的相机都使用相同的电平标准，均为 1.8~2.5V，小于 20mA 的电源输入，触发方式为下降沿触发，SMA 接口内芯为正，外围铜口为 GND。另外，有些使用 SMA 触发接口的相机只有一个 Trigger In 接口，没有 Trigger Out 接口，因此只能使用 Trigger In 功能。
https://note.youdao.com/share/?token=B19CB8A8576141279B0AB6E16913792D&gid=7234866

50.设置自动曝光功能

启用此功能后，SDK 会根据设置的阈值自动调节相机的曝光时间，使图像亮度在阈值上下波动，需要注意的是开启自动曝光功能之后不能手动设置曝光时间，否则会对功能产生影响，目前此功能为测试功能，可能会有使用上或者稳定性的问题，并且只支持 8 位模式。

自动曝光有五个可设置模式，分别为关闭自动曝光、仅调节 gain 模式、仅调节 exp 模式、混合调节模式、全天模式(暂未实现)，设置自动曝光模式的函数为 SetQHYCCDParam(handle, CONTROL_AUTOEXPOSURE, value)，各个模式对应的参数值为 0、1、2、3、4。

另外可以根据需要设置遮罩模式，可设置的遮罩模式有三个，分别为无遮罩图、遮罩图 A(中心位置，占比 21.5%左右，适合拍摄较大目标，如月亮)、遮罩图 B(中心位置，占比 5%左右，适合拍摄小目标，如星点)，设置遮罩模式的函数为 SetQHYCCDParam(handle, CONTROL_AUTOEXPmessureMethod, value)，各个模式对应的参数值为 0、1、2。

相机初始化之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep、SetQHYCCDParam，下为设置此功能的示例代码：

```
uit32_t ret = QHYCCD_ERROR;
double min = 0.0, max = 0.0, step = 0.0;
double value = 0.0;

//调用前判断相机是否具有该功能
ret = IsQHYCCDControlAvailable(handle, CONTROL_AUTOEXPOSURE);
if(ret == QHYCCD_SUCCESS)
{
    printf("call function success.\n");
}
//获取自定义功能的最大最小设置值以及设置步长
ret = GetQHYCCDParamMinMaxStep(handle, CONTROL_AUTOEXPOSURE, &min, &max, &step);
if(ret == QHYCCD_SUCCESS)
{
    printf("call function success.\n");
}
//设置自定义测试功能值
ret = SetQHYCCDParam(handle, CONTROL_AUTOEXPOSURE, 1.0);
if(ret == QHYCCD_SUCCESS)
{
    printf("call function success.\n");
}
//设置自动曝光阈值
ret = SetQHYCCDParam(handle, CONTROL_AUTOEXPmessureValue, 100.0);
if(ret == QHYCCD_SUCCESS)
```

```
{
	printf("call function success.\n");
}
//设置自动曝光的遮罩模式
ret = SetQHYCCDParam(handle, CONTROL_AUTOEXPmessureMethod, 0.0);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}
//获取自定义测试功能的当前值
value = GetQHYCCDParam(handle, CONTROL_AUTOEXPOSURE);
printf("value = %f\n", value);
```

51.设置自动白平衡功能

启用此功能后，SDK 会自动设置参数以调节图像白平衡，目前此功能为测试功能，可能会有使用上或稳定性的问题，并且目前只支持 8 位图像。此外，打开自动白平衡之后只能执行 15 次，之后会结束，想要执行需再次开启。

相机初始化之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、GetQHYCCDParamMinMaxStep、SetQHYCCDParam，下为设置此功能的示例代码：

```
uint32_t ret = QHYCCD_ERROR;
double min = 0, max = 0, step = 0;
double value = 0.0;

//调用前判断相机是否具有该功能
ret = IsQHYCCDControlAvailable(handle, CONTROL_AUTOWHITEBALANCE);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}

//获取自定义功能的最大最小设置值以及设置步长
ret = GetQHYCCDParamMinMaxStep(handle, CONTROL_AUTPWHITEBALANCE, &min, &max, &step);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}

//设置自定义测试功能值
ret = SetQHYCCDParam(handle, CONTROL_AUTOWHITEBALANCE, 1.0);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}

//获取自定义测试功能的当前值
value = GetQHYCCDParam(handle, CONTROL_AUTOWHITEBALANCE);
printf("value = %f\n", value);
```

52.设置帧监测功能

此功能可以用来监测图像数据是否在传输过程中发生意外，从而导致图像错位变形等问题，开启此功能后 SDK 会根据设置在指定位置检测数据校验值，校验不通过则判定为图像异常，将自动过滤此图像。

相机初始化之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、SetQHYCCDFrameDetectOnOff、SetQHYCCDFrameDetectCode、SetQHYCCDFrameDetectPos，下为设置此功能的示例代码：

```
uint32_t ret = QHYCCD_ERROR;

...
//连接相机并设置必要参数
...
//检查相机是否具有帧监测功能
ret = IsQHYCCDControlAvailable(handle, CONTROL_FrameDetect);
if(ret == QHYCCD_SUCCESS)
{
	printf("Camera has frame detect function\n");
}

//设置帧监测开启
ret = SetQHYCCDFrameDetectOnOff(handle, true);
if(ret == QHYCCD_SUCCESS)
{
	printf(“call function success.\n”);
}

//设置帧监测校验值
ret = SetQHYCCDFrameDetectCode(handle, 0);
if(ret == QHYCCD_SUCCESS)
{
	printf(“call function success.\n”);
}

//设置帧监测校验数据位置
ret = SetQHYCCDFrameDetectPos(handle, 0);
if(ret == QHYCCD_SUCCESS)
{
	printf(“call function success.\n”);
}
...
//获取连续图像
...
```

53.图像稳像功能

启用此功能可以稳定图像，主要用于追踪星点、空间站等移动目标，通常用于连续模式下，此功能为软件功能，根据算法使追踪目标始终处于图像中心。目前此功能为测试功能，可能会有使用上或者稳定性的问题。

相机初始化之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、SetQHYCCDParam、

GetQHYCCDImageStabilityGravity，下为设置此功能的示例代码：

```
uint32_t ret = QHYCCD_ERROR;
int GravityX = 0, GravityY = 0;

//检查是否具有稳像功能
ret = IsQHYCCDControlAvailable(handle, CONTROL_ImageStabilization);
if(ret == QHYCCD_SUCCESS)
{
	printf("Camera has this function.\n");
}

//设置稳像功能开启
ret = SetQHYCCDParam(handle, CONTROL_ImageStabilization, 1.0);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}
```

54.增益换算功能

使用此功能可以根据 SDK 的增益设置值转换成 dB 增益值，或反过来将 dB 增益值转换成 SDK 增益设置值，目前此功能为测试功能，可能有使用上或稳定性的问题。

相机初始化之后可以使用此功能，使用的函数有 IsQHYCCDContronAvailable、QHYCCD_DbGainToGainValue、QHYCCD_GainValueToDbGain，下为设置此功能的示例代码：

```
uint32_t ret = QHYCCD_ERROR;
double value = 0.0;

//检查是否具有增益转换功能
ret = IsQHYCCDControlAvailable(handle, CAM_GainDBConversion);
if(ret == QHYCCD_SUCCESS)
{
	printf("Camera has this function.\n");
}

//将 dB 增益转换成 SDK 增益值
ret = QHYCCD_DbGainToGainValue(handle, 100.0, &value);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}

//将 SDK 增益值转换成 dB 增益值
ret = QHYCCD_GainValueToDbGain(handle, 100.0, &value);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}
```

55.设置 dB 增益

使用此功能可以以 dB 为单位设置增益。

相机初始化之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、QHYCCD_DbGainToGainValue 和 SetQHYCCDParam，下为增益设置的示例代码：

```
int retVal = QHYCCD_ERROR;
double min, max, step;
double value = 0.0;

//检查是否具有增益转换功能
ret = IsQHYCCDControlAvailable(handle, CAM_GainDBConversion);
if(ret == QHYCCD_SUCCESS)
{
	printf("Camera has this function.\n");

	//将 dB 增益转换成 SDK 增益值
	ret = QHYCCD_DbGainToGainValue(handle, 100.0, &value);
	if(ret == QHYCCD_SUCCESS)
	{
		printf("call function success.\n");
	}

	//设置增益
	retVal = SetQHYCCDParam(camHandle, CONTROL_GAINdB, value);
	if(retVal == QHYCCD_SUCCESS)
	{
		printf("Set gain successfully.\n");
	}
}
```

56.获取系统增益、满井、读出噪声曲线值功能

使用此功能可以根据 SDK 增益设置值获取系统增益、满井、读出噪声曲线中对应的值。

相机初始化之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、QHYCCD_curveFullWell、QHYCCD_curveReadoutNoise、QHYCCD_curveSystemGain，下为设置此功能的示例代码：

```
uint32_t ret = QHYCCD_ERROR;
double value = 0.0;

//检查相机是否具有获取系统增益曲线值功能
ret = IsQHYCCDControlAvailable(handle, CAM_CurveSystemGain);
if(ret == QHYCCD_SUCCESS)
{
	printf("Camera has this function.\n");
}
//获取系统增益曲线值
ret = QHYCCD_curveSystemGain(handle, 100.0, &value);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}

//检查相机是否具有获取满井曲线值功能
```

```
ret = IsQHYCCDControlAvailable(handle, CAM_CurveFullWell);
if(ret == QHYCCD_SUCCESS)
{
	printf("Camera has this function.\n");
}
//获取满井曲线值
ret = QHYCCD_curveFullWell(handle, 100.0, &value);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}

//检查相机是否具有获取读出噪声曲线值功能
ret = IsQHYCCDControlAvailable(handle, CAM_CurveReadoutNoise);
if(ret == QHYCCD_SUCCESS)
{
	printf("Camera has this function.\n");
}
//获取读出噪声曲线值
ret = QHYCCD_curveReadoutNoise(handle, 100.0, &value);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}
```

57.设置图像镜像或旋转

使用此功能可以对单帧或连续模式的图像执行镜像或旋转操作，可执行操作包括旋转 180 度、向左旋转 90 度、向右旋转 90 度、水平方向镜像、垂直方向镜像，对用的参数值分别为 1、2、3、4。需要注意的是，此功能为图像数据的后期处理，而非对相机进行参数设置。

相机初始化之后可以使用此功能，设置此功能后，使用 GetQHYCCDSingleFrame 或 GetQHYCCDLiveFrame 函数获取图像时就会直接获得处理过的图像数据，使用到的函数有 IsQHYCCDControlAvailable、SetQHYCCDParam，下为设置此功能的示例代码：

```
uint32_t ret = QHYCCD_ERROR;

...
//连接相机并设置必要参数
...
//检查相机是否具有此功能
ret = IsQHYCCDControlAvailable(handle, CONTROL_ImgProc);
if(ret == QHYCCD_SUCCESS)
{
	printf("call function success.\n");
}

//设置为旋转 180 度
ret = SetQHYCCDParam(handle, CONTROL_ImgProc, 1.0);
if(ret == QHYCCD_SUCCESS)
{
```

```
        printf(“set image process success.\n”);
}
...
//获取图像
...
```

58.全局复位

此功能仅对滚动快门相机起效，作用为更改相机的曝光方式，使滚动快门相机由原来的逐行曝光方式变成所有行同时曝光。需要注意的是，此功能仅设置所有行一起曝光，数据还是按照逐行读出的方式，因此启用此功能后每行的曝光时间是不同的，第一行最短，向下逐渐增加，最后会导致图像底部比顶部更亮的情况。

相机初始化之后可以设置此功能，单帧连续模式下都支持此功能，使用的函数有IsQHYCCDControlAvailable、SetQHYCCDParam，下为设置此功能的示例代码：

```
uint32_t ret = QHYCCD_ERROR;

...
//连接相机并设置必要参数
...
//检查是否具有全局复位功能
ret = IsQHYCCDControlAvailable(handle, CONTROL_GlobalReset);
if(ret == QHYCCD_SUCCESS)
{
    //设置全局复位功能开启
    ret = SetQHYCCDParam(handle, CONTROL_GlobalReset, 1.0);
    if(ret == QHYCCD_SUCCESS)
    {
        printf("Set global reset success\n");
    }
}
...
//获取图像
...
```

59.残影消除功能

开启此功能可以消除图像中的残影。

相机初始化之后可以设置此功能，单帧连续模式下都支持此功能，使用的函数有IsQHYCCDControlAvailable、SetQHYCCDParam，下为设置此功能的示例代码：

```
uint32_t ret = QHYCCD_ERROR;

...
//连接相机并设置必要参数
...
//检查是否具有残影消除功能
ret = IsQHYCCDControlAvailable(handle, CONTROL_RemoveRBI);
if(ret == QHYCCD_SUCCESS)
{
```

```
    //设置残影消除功能开启
    ret = SetQHYCCDParam(handle, CONTROL_RemoveRBI, 1.0);
    if(ret == QHYCCD_SUCCESS)
    {
        printf("Set Remove RBI success\n");
    }
}
...
//获取图像
...
```

60.去除热噪声功能

开启此功能可以去除图像中的部分热噪声，此功能为软件功能，通过算法实现去除热噪声功能，需要注意的是此功能需要设置阈值，若阈值设置得过小会将星点也当做是热噪点去掉，若阈值设置得过大则消除热噪声效果不明显，需要自己根据实际情况调节阈值以达到最佳效果。另外，此功能为测试功能，可能会有使用上或稳定性方面的问题。

初始化相机之后可以使用此功能，使用的函数有 IsQHYCCDControlAvailable、SetQHYCCDParam，下为设置此功能的示例代码：

```
uint32_t ret = QHYCCD_ERROR;

...
//连接相机并设置必要参数
...
//检查是否具有去除热噪声功能
ret = IsQHYCCDControlAvailable(handle, CONTROL_DPC);
if(ret == QHYCCD_SUCCESS)
{
    //设置去除热噪声功能开启
    ret = SetQHYCCDParam(handle, CONTROL_DPC, 1.0);
    if(ret == QHYCCD_SUCCESS)
    {
        printf("Set Remove RBI success\n");
    }

    //设置阈值，默认阈值为 40
    ret = SetQHYCCDParam(handle, CONTROL_DPC_value, 30.0);
    if(ret == QHYCCD_SUCCESS)
    {
        printf("Set Remove RBI success\n");
    }
}
...
//获取图像
...
```

四、 示例程序

1.单帧模式

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "qhyccd.h"

int main(int argc,char *argv[])
{
	int num = 0;
	qhyccd_handle *camhandle = NULL;
	int ret = QHYCCD_ERROR;
	char id[32];
	int found = 0;
	unsigned int w,h,bpp,channels;
	unsigned char *ImgData;
	double chipw,chiph,pixelw,pixelh;


	//初始化 SDK
	ret = InitQHYCCDResource();
	if(ret == QHYCCD_ERROR)
	{
		printf("Init SDK failed!\n");
		goto failure;
	}


	//扫描相机
	num = ScanQHYCCD();
	if(num <= 0)
	{
		printf("Not Found QHYCCD camera.\n\n");
		goto failure;
	}


	//获取相机 ID
	for(int i = 0;i < num;i++)
	{
```

```
        ret = GetQHYCCDId(i,id);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("connected a camera,id is %s\n",id);
            found = 1;
            break;
        }
    }

    if(found == 1)
    {
        //打开相机，获取设备句柄
        camhandle = OpenQHYCCD(id);
        if(camhandle == NULL)
        {
            printf("Open QHYCCD fail \n");
            goto failure;
        }

        //设置读出模式
        ret = SetQHYCCDReadMode(camhandle, 0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set read mode failed.\n");
            goto failure;
        }

        //设置单帧模式
        ret = SetQHYCCDStreamMode(camhandle, 0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set stream mode failed.\n");
            goto failure;
        }

        //初始化相机
        ret = InitQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Init camera failed.\n",ret);
```

```
        goto failure;
}

//获取相机硬件信息
ret = GetQHYCCDChipInfo(camhandle,&chipw,&chiph,&w,&h,&pixelw,&pixelh,&bpp);
if(ret == QHYCCD_SUCCESS)
{
        printf("GetQHYCCDChipInfo success!\n");
        printf("CCD/CMOS chip information:\n");
        printf("Chip width %3f mm,Chip height %3f mm\n",chipw,chiph);
        printf("Chip pixel width %3f um,Chip pixel height %3f um\n",pixelw,pixelh);
        printf("Chip Max Resolution is %d x %d,depth is %d\n",w,h,bpp);
}
else
{
        printf("GetQHYCCDChipInfo fail\n");
        goto failure;
}

//设置 BIN
ret = SetQHYCCDBinMode(camhandle, cambinx, cambiny);
if(ret == QHYCCD_ERROR)
{
        printf("Set BIN mode failed.\n");
        goto failure;
}

//设置分辨率
ret = SetQHYCCDResolution(camhandle, 0, 0, w, h);
if(ret == QHYCCD_ERROR)
{
        printf("Set resolution failed.\n");
        goto failure;
}

//设置 Color
ret = IsQHYCCDControlAvailable(camhandle,CAM_IS_COLOR);
if(ret == QHYCCD_SUCCESS)
{
        printf("This is a color camera.\n");
```

```
        ret = SetQHYCCDDebayerOnOff(camhandle,false);
        if(ret == QHYCCD_ERROR)
        {
            printf("Setup color failed.\n");
            goto failure;
        }
}

//设置 Bits
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_TRANSFERBIT);
if(ret == QHYCCD_SUCCESS)
{
        ret = SetQHYCCDParam(camhandle, CONTROL_TRANSFERBIT, 16);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set bits failed.\n");
            goto failure;
        }
}

//设置 Traffic
ret = IsQHYCCDControlAvailable(camhandle, CONTROL_USBTRAFFIC);
if(ret == QHYCCD_SUCCESS)
{
        ret = SetQHYCCDParam(camhandle, CONTROL_USBTRAFFIC, 30);
        if(ret == QHYCCD_ERROR)
        {
            printf("SetQHYCCDParam CONTROL_USBTRAFFIC failed.\n");
            goto failure;
        }
}

//设置 Gain
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
if(ret == QHYCCD_SUCCESS)
{
        ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,30);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set gain failed.\n");
            goto failure;
```

```
        }
    }

    //设置 Offset
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,10);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set offset failed.\n");
            goto failure;
        }
    }

    //设置 DDR
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_DDR);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_DDR,1.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set DDR failed.\n");
            goto failure;
        }
    }

    //设置曝光时间
    ret = SetQHYCCDParam(camhandle, CONTROL_EXPOSURE, 1000000);
    if(ret == QHYCCD_ERROR)
    {
        printf("Set expose time failed.\n");
        goto failure;
    }

    //获取图像数据内存长度，开辟内存空间
    uint32_t length = GetQHYCCDMemLength(camhandle);
    if(length > 0)
    {
        ImgData = (unsigned char *)malloc(length);
```

```
            memset(ImgData,0,length);
        }
        else
        {
            printf("Get the min memory space length failure \n");
            goto failure;
        }

        //开始曝光
        ret = ExpQHYCCDSingleFrame(camhandle);
        if( ret == QHYCCD_ERROR )
        {
            printf("Start single expose failed.\n");
            goto failure;
        }

        //获取图像数据
        ret = GetQHYCCDSingleFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Get single frame succeessfully.\n");
            //show the image,this function need you do in software
        }
        else
        {
            printf("Get single frame failed.\n",ret);
        }

        delete(ImgData);
}
else
{
        printf("The camera is not QHYCCD camera or other error. \n");
        goto failure;
}

if(camhandle)
{
        //结束相机拍摄
        ret = CancelQHYCCDExposingAndReadout(camhandle);
        if(ret == QHYCCD_ERROR)
```

```
        {
            printf("Cancel exposing and readout failed.\n");
            goto failure;
        }

        //关闭相机
        ret = CloseQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Close camera failed.\n");
            goto failure;
        }
    }

    //释放 SDK 资源
    ret = ReleaseQHYCCDResource();
    if(ret == QHYCCD_ERROR)
    {
        printf("Rlease SDK resource failed.\n");
        goto failure;
    }

    return 0;

failure:
    printf("Some fatal error happened.\n");
    return 1;
}
```

2.连续模式

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "qhyccd.h"

int main(int argc,char *argv[])
{
    int num = 0;
    qhyccd_handle *camhandle = NULL;
    int ret = QHYCCD_ERROR;
```

```
char id[32];
int found = 0;
unsigned int w,h,bpp,channels;
unsigned char *ImgData;
double chipw,chiph,pixelw,pixelh;

//初始化 SDK
ret = InitQHYCCDResource();
if(ret == QHYCCD_ERROR)
{
	printf("Init SDK failed!\n");
	goto failure;
}

//扫描相机
num = ScanQHYCCD();
if(num <= 0)
{
	printf("Not Found QHYCCD camera.\n\n");
	goto failure;
}

//获取相机 ID
for(int i = 0;i < num;i++)
{
	ret = GetQHYCCDId(i,id);
	if(ret == QHYCCD_SUCCESS)
	{
		printf("connected a camera,id is %s\n",id);
		found = 1;
		break;
	}
}

if(found == 1)
{
	//打开相机，获取设备句柄
	camhandle = OpenQHYCCD(id);
	if(camhandle == NULL)
	{
```

```
        printf("Open QHYCCD fail \n");
        goto failure;
}

//设置读出模式
ret = SetQHYCCDReadMode(camhandle, 0);
if(ret == QHYCCD_ERROR)
{
        printf("Set read mode failed.\n");
        goto failure;
}

//设置连续模式
ret = SetQHYCCDStreamMode(camhandle, 1);
if(ret == QHYCCD_ERROR)
{
        printf("Set stream mode failed.\n");
        goto failure;
}

//初始化相机
ret = InitQHYCCD(camhandle);
if(ret == QHYCCD_ERROR)
{
        printf("Init camera failed.\n",ret);
        goto failure;
}

//获取相机硬件信息
ret = GetQHYCCDChipInfo(camhandle,&chipw,&chiph,&w,&h,&pixelw,&pixelh,&bpp);
if(ret == QHYCCD_SUCCESS)
{
        printf("GetQHYCCDChipInfo success!\n");
        printf("CCD/CMOS chip information:\n");
        printf("Chip width %3f mm,Chip height %3f mm\n",chipw,chiph);
        printf("Chip pixel width %3f um,Chip pixel height %3f um\n",pixelw,pixelh);
        printf("Chip Max Resolution is %d x %d,depth is %d\n",w,h,bpp);
}
else
{
```

```
        printf("GetQHYCCDChipInfo fail\n");
        goto failure;
}

//设置 BIN
ret = SetQHYCCDBinMode(camhandle, cambinx, cambiny);
if(ret == QHYCCD_ERROR)
{
        printf("Set BIN mode failed.\n");
        goto failure;
}

//设置分辨率
ret = SetQHYCCDResolution(camhandle, 0, 0, w, h);
if(ret == QHYCCD_ERROR)
{
        printf("Set resolution failed.\n");
        goto failure;
}

//设置 Color
ret = IsQHYCCDControlAvailable(camhandle,CAM_IS_COLOR);
if(ret == QHYCCD_SUCCESS)
{
        printf("This is a color camera.\n");
        ret = SetQHYCCDDebayerOnOff(camhandle,false);
        if(ret == QHYCCD_ERROR)
        {
                printf(“Setup color failed.\n”);
                goto failure;
        }
}

//设置 Bits
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_TRANSFERBIT);
if(ret == QHYCCD_SUCCESS)
{
        ret = SetQHYCCDParam(camhandle, CONTROL_TRANSFERBIT, 16);
        if(ret == QHYCCD_ERROR)
        {
```

```
            printf("Set bits failed.\n");
            goto failure;
        }
    }

    //设置 Gain
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,30);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set gain failed.\n");
            goto failure;
        }
    }

    //设置 Offset
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,10);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set offset failed.\n");
            goto failure;
        }
    }

    //设置 Traffic
    ret = IsQHYCCDControlAvailable(camhandle, CONTROL_USBTRAFFIC);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle, CONTROL_USBTRAFFIC, 30);
        if(ret == QHYCCD_ERROR)
        {
            printf("SetQHYCCDParam CONTROL_USBTRAFFIC failed.\n");
            goto failure;
        }
    }
```

```
//设置 red balance
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBR);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_WBR,30.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set red balance failed.\n");
		goto failure;
	}
}

//设置 green balance
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBG);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_WBG,30.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set green balance failed.\n");
		goto failure;
	}
}

//设置 blue balance
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBB);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_WBB,30.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set blue balance failed.\n");
		goto failure;
	}
}

//设置 brightness
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_BRIGHTNESS);
if(ret == QHYCCD_SUCCESS)
{
```

```
        ret = SetQHYCCDParam(camhandle,CONTROL_BRIGHTNESS,0.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set brightness failed.\n");
            goto failure;
        }
    }

    //设置 contrast
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_CONTRAST);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_CONTRAST,0.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set contrast failed.\n");
            goto failure;
        }
    }

    //设置 Gamma
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAMMA);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_GAMMA,1.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set gamma failed.\n");
            goto failure;
        }
    }

    //设置 DDR
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_DDR);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_DDR,1.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set DDR failed.\n");
            goto failure;
```

```
    }
}

//设置曝光时间
ret = SetQHYCCDParam(camhandle, CONTROL_EXPOSURE, 10000);
if(ret == QHYCCD_ERROR)
{
    printf("Set expose time failed.\n");
    goto failure;
}

//获取图像数据内存长度，开辟内存空间
uint32_t length = GetQHYCCDMemLength(camhandle);
if(length > 0)
{
    ImgData = (unsigned char *)malloc(length);
    memset(ImgData,0,length);
}
else
{
    printf("Get the min memory space length failure \n");
    goto failure;
}

//开始曝光
ret = BeginQHYCCDLive(camhandle);
if( ret == QHYCCD_ERROR )
{
    printf("Start live expose failed.\n");
    goto failure;
}

int fame = 0;
//获取图像数据
while(frame < 100)
{
    ret = QHYCCD_ERROR;
    while(ret != QHYCCD_SUCCESS)
    {
        ret = GetQHYCCDLiveFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
```

```
            }

            printf("Get live frame succeessfully.\n");
            frame ++;
            //show the image,this function need you do in software
        }

        delete(ImgData);
}
else
{
        printf("The camera is not QHYCCD camera or other error. \n");
        goto failure;
}

if(camhandle)
{
        //结束相机拍摄
        ret = StopQHYCCDLive(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Stop live capture failed.\n");
            goto failure;
        }

        //关闭相机
        ret = CloseQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Close camera failed.\n");
            goto failure;
        }
}

//释放 SDK 资源
ret = ReleaseQHYCCDResource();
if(ret == QHYCCD_ERROR)
{
        printf("Rlease SDK resource failed.\n");
        goto failure;
}
```

```
    return 0;

failure:
    printf("Some fatal error happened.\n");
    return 1;
}
```

3.连续模式切换读出模式、BIN 模式、数据格式

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "qhyccd.h"

int main(int argc,char *argv[])
{
    int num = 0;
    qhyccd_handle *camhandle = NULL;
    int ret = QHYCCD_ERROR;
    char id[32];
    int found = 0;
    int fame = 0;
    unsigned int w,h,bpp,channels;
    unsigned char *ImgData;
    double chipw,chiph,pixelw,pixelh;


    //初始化 SDK

    ret = InitQHYCCDResource();
    if(ret == QHYCCD_ERROR)
    {
        printf("Init SDK failed!\n");
        goto failure;
    }


    //扫描相机

    num = ScanQHYCCD();
    if(num <= 0)
    {
        printf("Not Found QHYCCD camera.\n\n");
        goto failure;
```

```
}

//获取相机 ID
for(int i = 0;i < num;i++)
{
	ret = GetQHYCCDId(i,id);
	if(ret == QHYCCD_SUCCESS)
	{
		printf("connected a camera,id is %s\n",id);
		found = 1;
		break;
	}
}

if(found == 1)
{
	//打开相机，获取设备句柄
	camhandle = OpenQHYCCD(id);
	if(camhandle == NULL)
	{
		printf("Open QHYCCD fail \n");
		goto failure;
	}

	//设置读出模式 0
	ret = SetQHYCCDReadMode(camhandle, 0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set read mode failed.\n");
		goto failure;
	}

	//设置连续模式
	ret = SetQHYCCDStreamMode(camhandle, 1);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set stream mode failed.\n");
		goto failure;
	}
```

```
//初始化相机
ret = InitQHYCCD(camhandle);
if(ret == QHYCCD_ERROR)
{
	printf("Init camera failed.\n",ret);
	goto failure;
}

//获取相机硬件信息
ret = GetQHYCCDChipInfo(camhandle,&chipw,&chiph,&w,&h,&pixelw,&pixelh,&bpp);
if(ret == QHYCCD_SUCCESS)
{
	printf("GetQHYCCDChipInfo success!\n");
	printf("CCD/CMOS chip information:\n");
	printf("Chip width %3f mm,Chip height %3f mm\n",chipw,chiph);
	printf("Chip pixel width %3f um,Chip pixel height %3f um\n",pixelw,pixelh);
	printf("Chip Max Resolution is %d x %d,depth is %d\n",w,h,bpp);
}
else
{
	printf("GetQHYCCDChipInfo fail\n");
	goto failure;
}

//设置 BIN 模式为 1X1
cambinx = 1;
cambiny = 1;
ret = SetQHYCCDBinMode(camhandle, cambinx, cambiny);
if(ret == QHYCCD_ERROR)
{
	printf("Set BIN mode failed.\n");
	goto failure;
}
ret = SetQHYCCDResolution(camhandle, 0, 0, w/cambinx, h/cambiny);
if(ret == QHYCCD_ERROR)
{
	printf("Set resolution failed.\n");
	goto failure;
}

//设置数据格式为 RAW16
```

```
ret = IsQHYCCDControlAvailable(camhandle,CAM_IS_COLOR);
if(ret == QHYCCD_SUCCESS)
{
	printf("This is a color camera.\n");
	ret = SetQHYCCDDebayerOnOff(camhandle,false);
	if(ret == QHYCCD_ERROR)
	{
		printf("Setup color failed.\n");
		goto failure;
	}
}
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_TRANSFERBIT);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle, CONTROL_TRANSFERBIT, 16);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set bits failed.\n");
		goto failure;
	}
}

//设置 Traffic
ret = IsQHYCCDControlAvailable(camhandle, CONTROL_USBTRAFFIC);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle, CONTROL_USBTRAFFIC, 30);
	if(ret == QHYCCD_ERROR)
	{
		printf("SetQHYCCDParam CONTROL_USBTRAFFIC failed.\n");
		goto failure;
	}
}

//设置 Gain
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,30);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set gain failed.\n");
```

```
            goto failure;
        }
    }

    //设置 Offset
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,10);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set offset failed.\n");
            goto failure;
        }
    }

    //设置 DDR
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_DDR);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_DDR,1.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set DDR failed.\n");
            goto failure;
        }
    }

    //设置曝光时间
    ret = SetQHYCCDParam(camhandle, CONTROL_EXPOSURE, 1000000);
    if(ret == QHYCCD_ERROR)
    {
        printf("Set expose time failed.\n");
        goto failure;
    }

    //获取图像数据内存长度，开辟内存空间
    uint32_t length = GetQHYCCDMemLength(camhandle);
    if(length > 0)
    {
```

```
        ImgData = (unsigned char *)malloc(length);
        memset(ImgData,0,length);
}
else
{
        printf("Get the min memory space length failure \n");
        goto failure;
}

//开始曝光
ret = BeginQHYCCDLive(camhandle);
if( ret == QHYCCD_ERROR )
{
        printf("Start live expose failed.\n");
        goto failure;
}

//获取图像数据
while(frame < 100)
{
        ret = QHYCCD_ERROR;
        while(ret != QHYCCD_SUCCESS)
        {
                ret = GetQHYCCDLiveFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
        }

        printf("Get live frame succeessfully.\n");
        frame ++;
        //show the image,this function need you do in software
}

//结束相机拍摄
ret = StopQHYCCDLive(camhandle);
if(ret == QHYCCD_ERROR)
{
        printf("Stop live capture failed.\n");
        goto failure;
}

/*****************************************************************************/
/***** 从读出模式开始重新设置读出模式、BIN 模式、数据格式，并重新设置相关参数 *****/
```

```
/*********************************************************************************************/

//重新设置为读出模式 1
ret = SetQHYCCDReadMode(camhandle, 1);
if(ret == QHYCCD_ERROR)
{
	printf("Set read mode failed.\n");
	goto failure;
}

//设置连续模式
ret = SetQHYCCDStreamMode(camhandle, 1);
if(ret == QHYCCD_ERROR)
{
	printf("Set stream mode failed.\n");
	goto failure;
}

//初始化相机
ret = InitQHYCCD(camhandle);
if(ret == QHYCCD_ERROR)
{
	printf("Init camera failed.\n",ret);
	goto failure;
}

//重新设置 BIN 模式为 2X2 BIN
cambinx = 2;
cambiny = 2;
ret = SetQHYCCDBinMode(camhandle, cambinx, cambiny);
if(ret == QHYCCD_ERROR)
{
	printf("Set BIN mode failed.\n");
	goto failure;
}
ret = SetQHYCCDResolution(camhandle, 0, 0, w/cambinx, h/cambiny);
if(ret == QHYCCD_ERROR)
{
	printf("Set resolution failed.\n");
	goto failure;
```

```
}

//重新设置数据格式为 RAW8
ret = IsQHYCCDControlAvailable(camhandle,CAM_IS_COLOR);
if(ret == QHYCCD_SUCCESS)
{
	printf("This is a color camera.\n");
	ret = SetQHYCCDDebayerOnOff(camhandle, false);
	if(ret == QHYCCD_ERROR)
	{
		printf("Setup color failed.\n");
		goto failure;
	}
}
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_TRANSFERBIT);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle, CONTROL_TRANSFERBIT, 8);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set bits failed.\n");
		goto failure;
	}
}

//设置 Gain
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,30);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set gain failed.\n");
		goto failure;
	}
}

//设置 Offset
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
if(ret == QHYCCD_SUCCESS)
{
```

```
        ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,10);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set offset failed.\n");
            goto failure;
        }
    }

    //设置 Traffic
    ret = IsQHYCCDControlAvailable(camhandle, CONTROL_USBTRAFFIC);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle, CONTROL_USBTRAFFIC, 30);
        if(ret == QHYCCD_ERROR)
        {
            printf("SetQHYCCDParam CONTROL_USBTRAFFIC failed.\n");
            goto failure;
        }
    }

    //设置 red balance
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBR);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_WBR,30.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set red balance failed.\n");
            goto failure;
        }
    }

    //设置 green balance
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBG);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_WBG,30.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set green balance failed.\n");
            goto failure;
```

```
    }
}

//设置 blue balance
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBB);
if(ret == QHYCCD_SUCCESS)
{
    ret = SetQHYCCDParam(camhandle,CONTROL_WBB,30.0);
    if(ret == QHYCCD_ERROR)
    {
        printf("Set blue balance failed.\n");
        goto failure;
    }
}

//设置 brightness
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_BRIGHTNESS);
if(ret == QHYCCD_SUCCESS)
{
    ret = SetQHYCCDParam(camhandle,CONTROL_BRIGHTNESS,0.0);
    if(ret == QHYCCD_ERROR)
    {
        printf("Set brightness failed.\n");
        goto failure;
    }
}

//设置 contrast
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_CONTRAST);
if(ret == QHYCCD_SUCCESS)
{
    ret = SetQHYCCDParam(camhandle,CONTROL_CONTRAST,0.0);
    if(ret == QHYCCD_ERROR)
    {
        printf("Set contrast failed.\n");
        goto failure;
    }
}

//设置 Gamma
```

```
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAMMA);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_GAMMA,1.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set gamma failed.\n");
		goto failure;
	}
}

//设置 DDR
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_DDR);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_DDR,1.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set DDR failed.\n");
		goto failure;
	}
}

//设置曝光时间
ret = SetQHYCCDParam(camhandle, CONTROL_EXPOSURE, 10000);
if(ret == QHYCCD_ERROR)
{
	printf("Set expose time failed.\n");
	goto failure;
}

//开始曝光
ret = BeginQHYCCDLive(camhandle);
if( ret == QHYCCD_ERROR )
{
	printf("Start live expose failed.\n");
	goto failure;
}

//获取图像数据
```

```
        while(frame < 100)
        {
            ret = QHYCCD_ERROR;
            while(ret != QHYCCD_SUCCESS)
            {
                ret = GetQHYCCDLiveFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
            }

            printf("Get live frame succeessfully.\n");
            frame ++;
            //show the image,this function need you do in software
        }

        //结束相机拍摄
        ret = StopQHYCCDLive(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Stop live capture failed.\n");
            goto failure;
        }

        delete(ImgData);
    }
    else
    {
        printf("The camera is not QHYCCD camera or other error. \n");
        goto failure;
    }

    if(camhandle)
    {
        //关闭相机
        ret = CloseQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Close camera failed.\n");
            goto failure;
        }

        //释放 SDK 资源
        ret = ReleaseQHYCCDResource();
```

```
            if(ret == QHYCCD_ERROR)
            {
            printf("Rlease SDK resource failed.\n");
            goto failure;
            }
    }

    return 0;

failure:
    printf("Some fatal error happened.\n");
    return 1;
}
```

4.相机制冷及湿度压力传感器

自动模式

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include "qhyccd.h"
qhyccd_handle *camhandle = NULL;

int main(int argc,char *argv[])
{
    uint32_t ret = QHYCCD_ERROR;
    int num = 0;
    int found = 0;
    char id[32];
    double nowTemp, nowPWM;
    int time = 0;


    ret = InitQHYCCDResource();
    if(ret == QHYCCD_ERROR)
    {
        printf("Initialize SDK resource failed.\n");
        goto failure;
    }

    num = ScanQHYCCD();
```

```
if(num <= 0)
{
	printf("Not Found QHYCCD,please check the usblink or the power.\n");
	goto failure;
}

for(int i = 0;i < num;i++)
{
	ret = GetQHYCCDId(i,id);
	if(ret == QHYCCD_SUCCESS)
	{
		printf("Connected to the first camera from the list,id is %s.\n",id);
		found = 1;
	}
}

if(found == 1)
{
	camhandle = OpenQHYCCD(id);
	if(camhandle == NULL)
	{
		printf("Open camera failed.\n");
		goto failure;
	}

	ret = SetQHYCCDStreamMode(camhandle,0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set stream mode failed.\n");
		goto failure;
	}

	ret = InitQHYCCD(camhandle);
	if(ret == QHYCCD_ERROR)
	{
		printf("Init camera failed.\n");
		goto failure;
	}

	/*********************************************************************************/

	/**********自动模式，此模式需要设置目标温度，CCD 相机需要持续设置**********/

	/**********CMOS 相机可以只设置一次，也可以持续设置多次，自动模式**********/
```

```
    /***********和手动模式不能同时使用                                          ************/
    /*****************************************************************************/
    //循环控制相机制冷 10min
    while(time < 600)
    {
        //设置目标温度为-10 度
        ret = SetQHYCCDParam(camhandle, CONTROL_COOLER, -10.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set target temperature failed.\n");
        }

        nowTemp = GetQHYCCDParam(camhandle,CONTROL_CURTEMP);//获取当前温度

        nowPWM = GetQHYCCDParam(camhandle,CONTROL_CURPWM);//获取当前功率

        printf("Temperature:%.1f°C,PWM:%.1f%%.\n",nowTemp,nowPWM/255.0 * 100);

        time ++;
        sleep(1000); //延时 1s
    }

    //关闭相机制冷
    ret = SetQHYCCDParam(camhandle, CONTROL_MANULPWM, 0);
    if(ret == QHYCCD_ERROR)
    {
        printf("Close camera cooler failed!(%d)\n",ret);
    }
}

if(camhandle)
{
    ret = CloseQHYCCD(camhandle);//关闭相机
    if(ret == QHYCCD_ERROR)
    {
        printf("Close camera failed.\n");
        goto failure;
```

```
            }


            ret = ReleaseQHYCCDResource();//释放相机资源
            if(ret == QHYCCD_ERROR)
            {
                printf("Rlease SDK resource failed.\n");
                goto failure;
            }
        }

        return 0;

failure:
        printf("some fatal error happened\n");
        return 1;
}
```

手动模式

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include "qhyccd.h"
qhyccd_handle *camhandle = NULL;

int main(int argc,char *argv[])
{
        int ret = QHYCCD_ERROR;
        int found = 0, num = 0;
        char id[32];

        ret = InitQHYCCDResource();
        if(ret == QHYCCD_ERROR)
        {
            printf("Initialize SDK resource failed.\n");
            goto failure;
        }

        num = ScanQHYCCD();
        if(num <= 0)
        {
            printf("Not Found QHYCCD,please check the usblink or the power.\n");
```

```
        goto failure;
}

for(int i = 0;i < num;i++)
{
        ret = GetQHYCCDId(i,id);
        if(ret == QHYCCD_SUCCESS)
        {
                printf("Connected to the first camera from the list,id is %s.\n",id);
                found = 1;
        }
}

if(found == 1)
{
        camhandle = OpenQHYCCD(id);
        if(camhandle == NULL)
        {
                printf("Open camera failed.\n");
                goto failure;
        }

        ret = SetQHYCCDStreamMode(camhandle,0);
        if(ret == QHYCCD_ERROR)
        {
                printf("Set stream mode failed.\n");
                goto failure;
        }

        ret = InitQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
                printf("Init camera failed.\n");
                goto failure;
        }

        int time = 0;
        double nowTemp, nowPWM;

        /*************************************************************************/

        /**********手动模式，此模式需要设置制冷功率，CCD 相机和 CMSO 相机都**********/

        /**********只需要设置一次，手动模式和自动模式不能同时使用********************/
```

```
    /*****************************************************************************/
    //设置制冷功率为 70%
    double pwm = 70.0;
    ret = SetQHYCCDParam(camhandle, CONTROL_MANULPWM, pwm / 100.0 * 255.0);
    if(ret == QHYCCD_ERROR)
    {
        printf("Set camera PWM failed.\n");
    }

    //持续获取相机温度 10min
    while(time < 600)
    {
        nowTemp = GetQHYCCDParam(camhandle,CONTROL_CURTEMP);//获取当前温度

        nowPWM = GetQHYCCDParam(camhandle,CONTROL_CURPWM);//获取当前功率

        printf("Temperature:%.1f°C,PWM:%.1f%%.\n",nowTemp,nowPWM/255.0 * 100);


        time ++;

        sleep(1000); //延时 1s

    }

    //关闭相机制冷
    ret = SetQHYCCDParam(camhandle, CONTROL_MANULPWM, 0);
    if(ret == QHYCCD_ERROR)
    {
        printf("Close camera cooler failed!(%d)\n",ret);
    }
}
if(camhandle)
{
    ret = CloseQHYCCD(camhandle);//关闭相机
    if(ret == QHYCCD_ERROR)
    {
        printf("Close QHYCCD failed.\n");
        goto failure;
    }
```

```
        ret = ReleaseQHYCCDResource();//释放相机资源
        if(ret == QHYCCD_ERROR)
        {
            printf("Rlease SDK resource failed.\n");
            goto failure;
        }
    }

    return 0;

failure:
    printf("some fatal error happened\n");
    return 1;
}
```

5.滤镜轮控制

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include "qhyccd.h"
qhyccd_handle *camhandle = NULL;

int main(int argc,char *argv[])
{
    int ret = QHYCCD_ERROR;
    int found = 0, num = 0;
    char id[32];
    double status;

    ret = InitQHYCCDResource();//初始化 SDK 资源
    if(ret == QHYCCD_ERROR)
    {
        printf("Initialize SDK resource failed.\n");
        goto failure;
    }

    num = ScanQHYCCD();//扫描相机
    if(num <= 0)
```

```
{
	printf("Not Found QHYCCD,please check the usblink or the power.\n");
	goto failure;
}

for(int i = 0;i < num;i++)
{
	ret = GetQHYCCDId(i,id);//获取相机 ID
	if(ret == QHYCCD_SUCCESS)
	{
		printf("Connected to the first camera from the list,id is %s.\n",id);
		found = 1;
	}
}

if(found == 1)
{
	camhandle = OpenQHYCCD(id);//打开相机
	if(camhandle == NULL)
	{
		printf("Open camera failed.\n");
		goto failure;
	}

	ret = SetQHYCCDReadMode(camhandle,0);//设置读出模式
	if(ret == QHYCCD_ERROR)
	{
		printf("Set read mode failed.\n");
		goto failure;
	}

	ret = SetQHYCCDStreamMode(camhandle,0);//设置为单帧模式
	if(ret == QHYCCD_ERROR)
	{
		printf("Set stream mode failed.\n");
		goto failure;
	}

	ret = InitQHYCCD(camhandle);//初始化相机
```

```
    if(ret == QHYCCD_ERROR)
    {
        printf("Init camera failed.\n");
        goto failure;
    }

    ret = IsQHYCCDCFWPlugged(camhandle);//检查滤镜轮连接状态
    if(ret == QHYCCD_SUCCESS)
    {
        printf("CFW is plugged.\n");

        num = (int)GetQHYCCDParam(camhandle, CONTROL_SLOTSNUM);//获取滤镜轮孔数
        if(num != 0)
        {
            ret = SetQHYCCDParam(camhandle, CONTROL_CFWPORT, 50.0);//设置目标孔位为第三孔
            if(ret == QHYCCD_SUCCESS)
            {
                while(status != 50.0)//循环获取位置，判断是否转到目标位置
                {
                    status = GetQHYCCDParam(camHandle, CONTROL_CFWPORT);//获取当前位置

                    sleep(500);//延时 500ms
                }
            }
        }
    }
}

if(camhandle)
{
    ret = CloseQHYCCD(camhandle);//关闭相机
    if(ret == QHYCCD_ERROR)
    {
        printf("Close QHYCCD failed.\n");
        goto failure;
    }

    ret = ReleaseQHYCCDResource();//释放相机资源
```

```
            if(ret == QHYCCD_ERROR)
            {
                printf("Rlease SDK resource failed.\n");
                goto failure;
            }
        }

        return 0;

failure:
        printf("some fatal error happened\n");
        return 1;
}
```

6.GPS 功能

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "qhyccd.h"

int main(int argc,char *argv[])
{
        int num = 0;
        qhyccd_handle *camhandle = NULL;
        int ret = QHYCCD_ERROR;
        char id[32];
        int found = 0;
        unsigned int w,h,bpp,channels;
        unsigned char *ImgData;
        double chipw,chiph,pixelw,pixelh;
        unsigned char gps[44] = { 0 };
        int seqNumber, tempNumber, width, height;
        int status, pps;
        int temp, deg, min, south, west;
        double fractMin, latitude, longitude;
        int start_sec, start_us, end_sec, end_us, now_sec, now_us;
        double exposure;



        //初始化 SDK
        ret = InitQHYCCDResource();
        if(ret == QHYCCD_ERROR)
```

```
{
	printf("Init SDK failed!\n");
	goto failure;
}

//扫描相机
num = ScanQHYCCD();
if(num <= 0)
{
	printf("Not Found QHYCCD camera.\n\n");
	goto failure;
}

//获取相机 ID
for(int i = 0;i < num;i++)
{
	ret = GetQHYCCDId(i,id);
	if(ret == QHYCCD_SUCCESS)
	{
		printf("connected a camera,id is %s\n",id);
		found = 1;
		break;
	}
}

if(found == 1)
{
	//打开相机，获取设备句柄
	camhandle = OpenQHYCCD(id);
	if(camhandle == NULL)
	{
		printf("Open QHYCCD fail \n");
		goto failure;
	}

	//设置读出模式
	ret = SetQHYCCDReadMode(camhandle, 0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set read mode failed.\n");
```

```
        goto failure;
}

//设置连续模式
ret = SetQHYCCDStreamMode(camhandle, 1);
if(ret == QHYCCD_ERROR)
{
        printf("Set stream mode failed.\n");
        goto failure;
}

//初始化相机
ret = InitQHYCCD(camhandle);
if(ret == QHYCCD_ERROR)
{
        printf("Init camera failed.\n",ret);
        goto failure;
}

//获取相机硬件信息
ret = GetQHYCCDChipInfo(camhandle,&chipw,&chiph,&w,&h,&pixelw,&pixelh,&bpp);
if(ret == QHYCCD_SUCCESS)
{
        printf("GetQHYCCDChipInfo success!\n");
        printf("CCD/CMOS chip information:\n");
        printf("Chip width %3f mm,Chip height %3f mm\n",chipw,chiph);
        printf("Chip pixel width %3f um,Chip pixel height %3f um\n",pixelw,pixelh);
        printf("Chip Max Resolution is %d x %d,depth is %d\n",w,h,bpp);
}
else
{
        printf("GetQHYCCDChipInfo fail\n");
        goto failure;
}

//设置 BIN
ret = SetQHYCCDBinMode(camhandle, cambinx, cambiny);
if(ret == QHYCCD_ERROR)
{
        printf("Set BIN mode failed.\n");
```

```
        goto failure;
}

//设置分辨率
ret = SetQHYCCDResolution(camhandle, 0, 0, w, h);
if(ret == QHYCCD_ERROR)
{
        printf("Set resolution failed.\n");
        goto failure;
}

//设置 Color
ret = IsQHYCCDControlAvailable(camhandle,CAM_IS_COLOR);
if(ret == QHYCCD_SUCCESS)
{
        printf("This is a color camera.\n");
        ret = SetQHYCCDDebayerOnOff(camhandle,false);
        if(ret == QHYCCD_ERROR)
        {
                printf(“Setup color failed.\n”);
                goto failure;
        }
}

//设置 Bits
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_TRANSFERBIT);
if(ret == QHYCCD_SUCCESS)
{
        ret = SetQHYCCDParam(camhandle, CONTROL_TRANSFERBIT, 16);
        if(ret == QHYCCD_ERROR)
        {
                printf("Set bits failed.\n");
                goto failure;
        }
}

//设置 Gain
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
if(ret == QHYCCD_SUCCESS)
{
```

```
        ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,30);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set gain failed.\n");
            goto failure;
        }
    }

    //设置 Offset
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,10);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set offset failed.\n");
            goto failure;
        }
    }

    //设置 Traffic
    ret = IsQHYCCDControlAvailable(camhandle, CONTROL_USBTRAFFIC);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle, CONTROL_USBTRAFFIC, 30);
        if(ret == QHYCCD_ERROR)
        {
            printf("SetQHYCCDParam CONTROL_USBTRAFFIC failed.\n");
            goto failure;
        }
    }

    //设置 red balance
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBR);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_WBR,30.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set red balance failed.\n");
            goto failure;
```

```
        }
    }

    //设置 green balance
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBG);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_WBG,30.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set green balance failed.\n");
            goto failure;
        }
    }

    //设置 blue balance
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBB);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_WBB,30.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set blue balance failed.\n");
            goto failure;
        }
    }

    //设置 brightness
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_BRIGHTNESS);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_BRIGHTNESS,0.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set brightness failed.\n");
            goto failure;
        }
    }

    //设置 contrast
```

```
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_CONTRAST);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_CONTRAST,0.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set contrast failed.\n");
		goto failure;
	}
}

//设置 Gamma
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAMMA);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_GAMMA,1.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set gamma failed.\n");
		goto failure;
	}
}

//设置 DDR
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_DDR);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_DDR,1.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set DDR failed.\n");
		goto failure;
	}
}

//设置曝光时间
ret = SetQHYCCDParam(camhandle, CONTROL_EXPOSURE, 10000);
if(ret == QHYCCD_ERROR)
{
	printf("Set expose time failed.\n");
	goto failure;
```

```
}

//获取图像数据内存长度，开辟内存空间
uint32_t length = GetQHYCCDMemLength(camhandle);
if(length > 0)
{
    ImgData = (unsigned char *)malloc(length);
    memset(ImgData,0,length);
}
else
{
    printf("Get the min memory space length failure \n");
    goto failure;
}

//检查是否支持 GPS 功能
ret = IsQHYCCDControlAvailable(camHandle, CAM_GPS);
if(ret == QHYCCD_SUCCESS)
{
    printf(“The camera can use GPS function.\n”);
}

//打开 GPS 功能
ret = SetQHYCCDParam(camHandle, CAM_GPS, 1.0);
if(ret == QHYCCD_SUCCESS)
{
    printf(“Turn GPS on successfully.\n”);
}

//此设置需要根据实际情况进行调节，使 PPS 计数值接近 1000000 即可，仅 QHY174-GPS 需要
ret = SetQHYCCDGPSVCOXFreq(camHandle, 2);
if(ret == QHYCCD_SUCCESS)
{
    printf(“Set GPS VCOX frequency successfully.\n”);
}

//设置校准脉冲的位置 A 和位置 B，需要根据实际情况设置，仅 QHY174-GPS 需要
SetQHYCCDGPSPOSA(camHandle, 0, pos, 40);
SetQHYCCDGPSPOSB(camHandle, 0, pos, 40);
```

```
//开始曝光
ret = BeginQHYCCDLive(camhandle);
if( ret == QHYCCD_ERROR )
{
	printf("Start live expose failed.\n");
	goto failure;
}

int fame = 0;
//获取图像数据
while(frame < 100)
{
	ret = QHYCCD_ERROR;
	while(ret != QHYCCD_SUCCESS)
	{
		ret = GetQHYCCDLiveFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
	}

	printf("Get live frame succeessfully.\n");
	frame ++;
	//show the image,this function need you do in software

	memcpy(gps, ImgData, 44);
	//GPS 状态
	now_flag = (gps[33] / 16 ) %4;
	//PPS 计数值
	pps = 256*256*gps[41] + 256*gps[42] + gps[43];
	//帧序号
	seqNumber = 256*256*256*gps[0] + 256*256*gps[1] + 256*gps[2] + gps[3];
	//图像宽度
	width = 256*gps[5] + gps[6];
	//图像高度
	height = 256*gps[7] + gps[8];
	//纬度
	temp = 256*256*256*gps[9] + 256*256*gps[10] + 256*gps[11] + gps[12];
	south = temp > 1000000000;
	deg = (temp % 1000000000) / 10000000;
	min = (temp % 10000000) / 100000;
	fractMin = (temp % 100000) / 100000.0;
	latitude = (deg + (min + fractMin) / 60.0) * (south==0 ? 1 : -1);
	//经度
	temp = 256*256*256*gps[13] + 256*256*gps[14] + 256*gps[15] + gps[16];
```

```
            west = temp > 1000000000;
            deg = (temp % 1000000000) / 1000000;
            min = (temp % 1000000) / 10000;
            fractMin = (temp % 10000) / 10000.0;
            longitude = (deg + (min + fractMin) / 60.0) * (west==0 ? 1 : -1);
            //快门开始时间
            start_sec = 256*256*256*gps[18] + 256*256*gps[19] + 256*gps[20] + gps[21];
            start_us = 256*256*gps[22] + 256*gps[23] + gps[24];
            //快门结束时间
            end_sec = 256*256*256*gps[26] + 256*256*gps[27] + 256*gps[28] + gps[29];
            end_us = 256*256*gps[30] + 256*gps[31] + gps[32];
            //当前时间
            now_sec = 256*256*256*gps[34] + 256*256*gps[35] + 256*gps[36] + gps[37];
            now_us = 256*256*gps[38] + 256*gps[39] + gps[40];
            //曝光时间
            exposure = (end_sec - start_sec) * 1000 * 1000 + (end_us - start_us);
        }

        delete(ImgData);
    }
    else
    {
        printf("The camera is not QHYCCD camera or other error. \n");
        goto failure;
    }

    if(camhandle)
    {
        //结束相机拍摄
        ret = StopQHYCCDLive(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Stop live capture failed.\n");
            goto failure;
        }

        //关闭相机
        ret = CloseQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Close camera failed.\n");
            goto failure;
        }
    }
```

```
    //释放 SDK 资源
    ret = ReleaseQHYCCDResource();
    if(ret == QHYCCD_ERROR)
    {
        printf("Rlease SDK resource failed.\n");
        goto failure;
    }

    return 0;

failure:
    printf("Some fatal error happened.\n");
    return 1;
}
```

7.AntiRBI 功能

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "qhyccd.h"

int main(int argc,char *argv[])
{
    int num = 0;
    qhyccd_handle *camhandle = NULL;
    int ret = QHYCCD_ERROR;
    char id[32];
    int found = 0;
    unsigned int w,h,bpp,channels;
    unsigned char *ImgData;
    double chipw,chiph,pixelw,pixelh;

    //初始化 SDK
    ret = InitQHYCCDResource();
    if(ret == QHYCCD_ERROR)
    {
        printf("Init SDK failed!\n");
        goto failure;
    }
```

```
//扫描相机
num = ScanQHYCCD();
if(num <= 0)
{
	printf("Not Found QHYCCD camera.\n\n");
	goto failure;
}

//获取相机 ID
for(int i = 0;i < num;i++)
{
	ret = GetQHYCCDId(i,id);
	if(ret == QHYCCD_SUCCESS)
	{
		printf("connected a camera,id is %s\n",id);
		found = 1;
		break;
	}
}

if(found == 1)
{
	//打开相机，获取设备句柄
	camhandle = OpenQHYCCD(id);
	if(camhandle == NULL)
	{
		printf("Open QHYCCD fail \n");
		goto failure;
	}

	//设置读出模式
	ret = SetQHYCCDReadMode(camhandle, 0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set read mode failed.\n");
		goto failure;
	}

	//设置连续模式
```

```
ret = SetQHYCCDStreamMode(camhandle, 1);
if(ret == QHYCCD_ERROR)
{
	printf("Set stream mode failed.\n");
	goto failure;
}

//初始化相机
ret = InitQHYCCD(camhandle);
if(ret == QHYCCD_ERROR)
{
	printf("Init camera failed.\n",ret);
	goto failure;
}

//获取相机硬件信息
ret = GetQHYCCDChipInfo(camhandle,&chipw,&chiph,&w,&h,&pixelw,&pixelh,&bpp);
if(ret == QHYCCD_SUCCESS)
{
	printf("GetQHYCCDChipInfo success!\n");
	printf("CCD/CMOS chip information:\n");
	printf("Chip width %3f mm,Chip height %3f mm\n",chipw,chiph);
	printf("Chip pixel width %3f um,Chip pixel height %3f um\n",pixelw,pixelh);
	printf("Chip Max Resolution is %d x %d,depth is %d\n",w,h,bpp);
}
else
{
	printf("GetQHYCCDChipInfo fail\n");
	goto failure;
}

//设置 BIN
ret = SetQHYCCDBinMode(camhandle, cambinx, cambiny);
if(ret == QHYCCD_ERROR)
{
	printf("Set BIN mode failed.\n");
	goto failure;
}

//设置分辨率
```

```
ret = SetQHYCCDResolution(camhandle, 0, 0, w, h);
if(ret == QHYCCD_ERROR)
{
	printf("Set resolution failed.\n");
	goto failure;
}

//设置 Color
ret = IsQHYCCDControlAvailable(camhandle,CAM_IS_COLOR);
if(ret == QHYCCD_SUCCESS)
{
	printf("This is a color camera.\n");
	ret = SetQHYCCDDebayerOnOff(camhandle,false);
	if(ret == QHYCCD_ERROR)
	{
		printf("Setup color failed.\n");
		goto failure;
	}
}

//设置 Bits
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_TRANSFERBIT);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle, CONTROL_TRANSFERBIT, 16);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set bits failed.\n");
		goto failure;
	}
}

//设置 Gain
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,30);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set gain failed.\n");
		goto failure;
```

```
        }
    }

    //设置 Offset
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,10);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set offset failed.\n");
            goto failure;
        }
    }

    //设置 Traffic
    ret = IsQHYCCDControlAvailable(camhandle, CONTROL_USBTRAFFIC);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle, CONTROL_USBTRAFFIC, 30);
        if(ret == QHYCCD_ERROR)
        {
            printf("SetQHYCCDParam CONTROL_USBTRAFFIC failed.\n");
            goto failure;
        }
    }

    //设置 red balance
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBR);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_WBR,30.0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set red balance failed.\n");
            goto failure;
        }
    }

    //设置 green balance
```

```
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBG);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_WBG,30.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set green balance failed.\n");
		goto failure;
	}
}

//设置 blue balance
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_WBB);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_WBB,30.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set blue balance failed.\n");
		goto failure;
	}
}

//设置 brightness
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_BRIGHTNESS);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_BRIGHTNESS,0.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set brightness failed.\n");
		goto failure;
	}
}

//设置 contrast
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_CONTRAST);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_CONTRAST,0.0);
	if(ret == QHYCCD_ERROR)
```

```
    {
        printf("Set contrast failed.\n");
        goto failure;
    }
}

//设置 Gamma
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAMMA);
if(ret == QHYCCD_SUCCESS)
{
    ret = SetQHYCCDParam(camhandle,CONTROL_GAMMA,1.0);
    if(ret == QHYCCD_ERROR)
    {
        printf("Set gamma failed.\n");
        goto failure;
    }
}

//设置 DDR
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_DDR);
if(ret == QHYCCD_SUCCESS)
{
    ret = SetQHYCCDParam(camhandle,CONTROL_DDR,1.0);
    if(ret == QHYCCD_ERROR)
    {
        printf("Set DDR failed.\n");
        goto failure;
    }
}

//设置曝光时间
ret = SetQHYCCDParam(camhandle, CONTROL_EXPOSURE, 10000);
if(ret == QHYCCD_ERROR)
{
    printf("Set expose time failed.\n");
    goto failure;
}

//开启 AntiRBI 模式
ret = SetQHYCCDEnableLiveModeAntiRBI(camHandle, 1);
if(ret == QHYCCD_SUCCESS)
{
```

```
            printf("Enable live mode AntiRBI successfully.\n");
        }

        //获取图像数据内存长度，开辟内存空间
        uint32_t length = GetQHYCCDMemLength(camhandle);
        if(length > 0)
        {
            ImgData = (unsigned char *)malloc(length);
            memset(ImgData,0,length);
        }
        else
        {
            printf("Get the min memory space length failure \n");
            goto failure;
        }

        //开始曝光
        ret = BeginQHYCCDLive(camhandle);
        if( ret == QHYCCD_ERROR )
        {
            printf("Start live expose failed.\n");
            goto failure;
        }

        int fame = 0;
        //获取图像数据
        while(frame < 100)
        {
            ret = QHYCCD_ERROR;
            while(ret != QHYCCD_SUCCESS)
            {
                ret = GetQHYCCDLiveFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
            }

            printf("Get live frame succeessfully.\n");
            frame ++;
            //show the image,this function need you do in software
        }

        delete(ImgData);
    }
    else
```

```
    {
        printf("The camera is not QHYCCD camera or other error. \n");
        goto failure;
    }

    if(camhandle)
    {
        //结束相机拍摄
        ret = StopQHYCCDLive(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Stop live capture failed.\n");
            goto failure;
        }

        //关闭相机
        ret = CloseQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Close camera failed.\n");
            goto failure;
        }
    }

    //释放 SDK 资源
    ret = ReleaseQHYCCDResource();
    if(ret == QHYCCD_ERROR)
    {
        printf("Rlease SDK resource failed.\n");
        goto failure;
    }

    return 0;

failure:
    printf("Some fatal error happened.\n");
    return 1;
}
```

8.Burst 功能

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <qhyccd.h>
#include <sys/time.h>

int main(int argc,char *argv[])
{
	int num = 0;
	qhyccd_handle *camhandle;
	int ret;
	char id[32];
	int length;
	int found = 0;
	unsigned int w,h,bpp,channels;
	unsigned int ex, ey, sizex, sizey;
	unsigned char *ImgData;
	double chipw,chiph,pixelw,pixelh;

	unsigned int imagew;
	unsigned int imageh;

	ret = InitQHYCCDResource();
	if(ret == QHYCCD_ERROR)
	{
		printf("InitQHYCCDResource failed.\n");
		return;
	}

	num = ScanQHYCCD();
	if(num <= 0)
	{
		printf("ScanQHYCCD no camera be found.\n");
		return;
	}

	for(int i = 0;i < num;i++)
	{
		ret = GetQHYCCDId(i,id);
		if(ret == QHYCCD_SUCCESS)
		{
			printf("GetQHYCCDId id = %s\n", id);
			found = 1;
```

```
            break;
        }
    }

    if(found == 1)
    {
        camhandle = OpenQHYCCD(id);
        if(camhandle == NULL)
        {
            printf("OpenQHYCCD failed.\n");
            return;
        }

        ret = SetQHYCCDReadMode(camhandle, 0);
        if(ret == QHYCCD_ERROR)
        {
            printf("SetQHYCCDReadMode failed.\n");
        }

        ret = SetQHYCCDStreamMode(camhandle, 1);
        if(ret == QHYCCD_ERROR)
        {
            printf("SetQHYCCDStreamMode failed.\n");
        }

        ret = InitQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("InitQHYCCD failed.\n");
        }

        ret = GetQHYCCDEffectiveArea(camhandle, &ex, &ey, &sizex, &sizey);
        if(ret == QHYCCD_ERROR)
        {
            printf("GetQHYCCDEffectiveArea failed.\n");
        }

        ret = GetQHYCCDChipInfo(camhandle, &chipw, &chiph, &imagew, &imageh, &pixelw, &pixelh, &bpp);
        if (ret == QHYCCD_SUCCESS)
        {
            printf("GetQHYCCDChipInfo:\n");
            printf("Chip    size width x height         : %.3f x %.3f [mm]\n", chipw, chiph);
            printf("Pixel size width x height          : %.3f x %.3f [um]\n", pixelw, pixelh);
            printf("Image size width x height          : %d x %d\n", imagew, imageh);
```

```
}
else
{
	printf("GetQHYCCDChipInfo failed.\n");
	return;
}

ret = SetQHYCCDBinMode(camhandle, 1, 1);
if(ret == QHYCCD_SUCCESS)
{
	printf("SetQHYCCDBinMode failed.\n");
}

ret = SetQHYCCDResolution(camhandle, 0, 0, imagew, imageh);
if(ret == QHYCCD_ERROR)
{
	printf("SetQHYCCDResolution failed.\n");
}

ret = SetQHYCCDDebayerOnOff(camhandle, false);
if(ret == QHYCCD_ERROR)
{
	printf("SetQHYCCDDebayerOnOff failed.\n");
}

ret = SetQHYCCDParam(camhandle,CONTROL_TRANSFERBIT,16);
if(ret == QHYCCD_ERROR)
{
	printf("SetQHYCCDParam BITS failed.\n");
}

ret = SetQHYCCDParam(camhandle,CONTROL_BRIGHTNESS,0);
if(ret == QHYCCD_ERROR)
{
	printf("SetQHYCCDParam BRIGHTNESS failed.\n");
}

ret = SetQHYCCDParam(camhandle,CONTROL_CONTRAST,0);
if(ret == QHYCCD_ERROR)
{
	printf("SetQHYCCDParam CONTRAST failed.\n");
}

ret = SetQHYCCDParam(camhandle,CONTROL_GAMMA,1.0);
```

```
	if(ret == QHYCCD_ERROR)
	{
		printf("SetQHYCCDParam GAMMA failed.\n");
	}

	ret = SetQHYCCDParam(camhandle, CONTROL_USBTRAFFIC, 0);
	if(ret == QHYCCD_ERROR)
	{
		printf("SetQHYCCDParam TRAFFIC failed.\n");
	}

	ret = SetQHYCCDParam(camhandle, CONTROL_DDR, 1.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("SetQHYCCDParam DDR failed.\n");
	}

	ret = SetQHYCCDParam(camhandle, CONTROL_EXPOSURE, 200000);
	if(ret == QHYCCD_ERROR)
	{
		printf("SetQHYCCDParam EXPOSURE failed.\n");
	}

	length = GetQHYCCDMemLength(camhandle);
	printf("GetQHYCCDMemLength length = %d\n", length);
	if(length > 0)
	{
		ImgData = (unsigned char *)malloc(length);
		memset(ImgData,0,length);
	}

	//begin live capture,it will start live thread in SDK
	ret = BeginQHYCCDLive(camhandle);
	printf("BeginQHYCCDLive ret = %d\n", ret);

	//Setup start and end frame,after setting this,camera will output middle frames when capture,for example setup 1 and 3,camera will output 2 frame
	ret = SetQHYCCDBurstModeStartEnd(camhandle, 1, 3);
	//Add patch data,it ca avoid data not enough that can't output image
	ret = SetQHYCCDBurstModePatchNumber(camhandle, 2000);

	//Enable Burst Mode
	ret = EnableQHYCCDBurstMode(camhandle, true);
	usleep(10000);
```

```
        //Start Burst Mode capture,it need add a delay between two functions
        ret = SetQHYCCDBurstIDLE(camhandle);
        usleep(20000);
        ret = ReleaseQHYCCDBurstIDLE(camhandle);

        //Get images
        ret = QHYCCD_ERROR;
        while(ret != QHYCCD_SUCCESS)
        {
            ret = GetQHYCCDLiveFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
            //printf("GetQHYCCDLiveFrame ret= %d\n", ret);
        }
        printf("w = %d h = %d bpp = %d channels = %d\n", w, h, bpp, channels);

        ret = StopQHYCCDLive(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("StopQHYCCDLive failed.\n");
        }
    }

    delete(ImgData);

    if(camhandle)
    {
        ret = CloseQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("CloseQHYCCD failed.\n");
        }
    }

    ret = ReleaseQHYCCDResource();
    if(ret == QHYCCD_ERROR)
    {
        printf("ReleaseQHYCCDResource failed.\n");
    }

    return 0;
}
```

9.外触发功能

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "qhyccd.h"

int main(int argc,char *argv[])
{
	int num = 0;
	qhyccd_handle *camhandle = NULL;
	int ret = QHYCCD_ERROR;
	char id[32];
	int found = 0;
	unsigned int w,h,bpp,channels;
	unsigned char *ImgData;
	double chipw,chiph,pixelw,pixelh;
	double min, max, step;

	//初始化 SDK
	ret = InitQHYCCDResource();
	if(ret == QHYCCD_ERROR)
	{
		printf("Init SDK failed!\n");
		goto failure;
	}

	//扫描相机
	num = ScanQHYCCD();
	if(num <= 0)
	{
		printf("Not Found QHYCCD camera.\n\n");
		goto failure;
	}

	//获取相机 ID
	for(int i = 0;i < num;i++)
	{
		ret = GetQHYCCDId(i,id);
		if(ret == QHYCCD_SUCCESS)
		{
			printf("connected a camera,id is %s\n",id);
			found = 1;
```

```
            break;
        }
    }

    if(found == 1)
    {
        //打开相机，获取设备句柄
        camhandle = OpenQHYCCD(id);
        if(camhandle == NULL)
        {
            printf("Open QHYCCD fail \n");
            goto failure;
        }

        //设置读出模式
        ret = SetQHYCCDReadMode(camhandle, 0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set read mode failed.\n");
            goto failure;
        }

        //设置单帧模式
        ret = SetQHYCCDStreamMode(camhandle, 0);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set stream mode failed.\n");
            goto failure;
        }

        //初始化相机
        ret = InitQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Init camera failed.\n",ret);
            goto failure;
        }

        //获取相机硬件信息
```

```
ret = GetQHYCCDChipInfo(camhandle,&chipw,&chiph,&w,&h,&pixelw,&pixelh,&bpp);
if(ret == QHYCCD_SUCCESS)
{
	printf("GetQHYCCDChipInfo success!\n");
	printf("CCD/CMOS chip information:\n");
	printf("Chip width %3f mm,Chip height %3f mm\n",chipw,chiph);
	printf("Chip pixel width %3f um,Chip pixel height %3f um\n",pixelw,pixelh);
	printf("Chip Max Resolution is %d x %d,depth is %d\n",w,h,bpp);
}
else
{
	printf("GetQHYCCDChipInfo fail\n");
	goto failure;
}

//设置 BIN
ret = SetQHYCCDBinMode(camhandle, cambinx, cambiny);
if(ret == QHYCCD_ERROR)
{
	printf("Set BIN mode failed.\n");
	goto failure;
}

//设置分辨率
ret = SetQHYCCDResolution(camhandle, 0, 0, w, h);
if(ret == QHYCCD_ERROR)
{
	printf("Set resolution failed.\n");
	goto failure;
}

//设置 Color
ret = IsQHYCCDControlAvailable(camhandle,CAM_IS_COLOR);
if(ret == QHYCCD_SUCCESS)
{
	printf("This is a color camera.\n");
	ret = SetQHYCCDDebayerOnOff(camhandle,false);
	if(ret == QHYCCD_ERROR)
	{
		printf("Setup color failed.\n");
		goto failure;
```

```
        }
    }

    //设置 Bits
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_TRANSFERBIT);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle, CONTROL_TRANSFERBIT, 16);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set bits failed.\n");
            goto failure;
        }
    }

    //设置 Traffic
    ret = IsQHYCCDControlAvailable(camhandle, CONTROL_USBTRAFFIC);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle, CONTROL_USBTRAFFIC, 30);
        if(ret == QHYCCD_ERROR)
        {
            printf("SetQHYCCDParam CONTROL_USBTRAFFIC failed.\n");
            goto failure;
        }
    }

    //设置 Gain
    ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
    if(ret == QHYCCD_SUCCESS)
    {
        ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,30);
        if(ret == QHYCCD_ERROR)
        {
            printf("Set gain failed.\n");
            goto failure;
        }
    }

    //设置 Offset
```

```
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,10);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set offset failed.\n");
		goto failure;
	}
}

//设置 DDR
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_DDR);
if(ret == QHYCCD_SUCCESS)
{
	ret = SetQHYCCDParam(camhandle,CONTROL_DDR,1.0);
	if(ret == QHYCCD_ERROR)
	{
		printf("Set DDR failed.\n");
		goto failure;
	}
}

//设置曝光时间
ret = SetQHYCCDParam(camhandle, CONTROL_EXPOSURE, 1000000);
if(ret == QHYCCD_ERROR)
{
	printf("Set expose time failed.\n");
	goto failure;
}

//获取图像数据内存长度，开辟内存空间
uint32_t length = GetQHYCCDMemLength(camhandle);
if(length > 0)
{
	ImgData = (unsigned char *)malloc(length);
	memset(ImgData,0,length);
}
else
{
	printf("Get the min memory space length failure \n");
```

```
        goto failure;
}

//检查相机是否支持触发功能
retVal = IsQHYCCDControlAvailable(camHandle, CAM_TRIGER_INTERFACE);
iif(retVal == QHYCCD_SUCCESS)
{
        printf("This camera can use trigger function.\n");
}

//获取触发接口的数量和名称
uint32_t num = 0;
retVal = GetQHYCCDTrigerInterfaceNumber(camhandle, &num);
if(num > 1)
{
        char name[40] = { 0 };
        for(int i = 0; i < num; i ++)
        {
                retVal = GetQHYCCDTrigerInterName(camhandle, i, name);
                if(retVal == QHYCCD_SUCCESS)
                {
                        printf("Get triger interface successfully.\n");
                }
        }
}

//设置使用默认触发接口 0
retVal = SetQHYCCDTrigerInterface(camhandle, 0);
if(retVal == QHYCCD_SUCCESS)
{
        printf("Set interface successfully.\n");
}

//使能触发功能
retVal = SetQHYCCDTrigerFunction(camHandle, true);
if(retVal == QHYCCD_SUCCESS)
{
        printf("Enable trigger mode successfully.\n");
}

//使能触发输出功能
retVal = EnableQHYCCDTrigerOut(camhandle);
if(retVal == QHYCCD_SUCCESS)
{
        printf("Enable trigger out successfully.\n");
}


//开始曝光

ret = ExpQHYCCDSingleFrame(camhandle);
if( ret == QHYCCD_ERROR )
```

```
        {
            printf("Start single expose failed.\n");
            goto failure;
        }

        //Now you need send a trigger signal to camera,this signal can generated by software or hardware

        //获取图像数据

        ret = GetQHYCCDSingleFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Get single frame succeessfully.\n");
            //show the image,this function need you do in software
        }
        else
        {
            printf("Get single frame failed.\n",ret);
        }

        delete(ImgData);
}
else
{
        printf("The camera is not QHYCCD camera or other error. \n");
        goto failure;
}

if(camhandle)
{
        //结束相机拍摄

        ret = CancelQHYCCDExposingAndReadout(camhandle);
        if(ret == QHYCCD_ERROR)
        {
            printf("Cancel exposing and readout failed.\n");
            goto failure;
        }

        //关闭相机

        ret = CloseQHYCCD(camhandle);
        if(ret == QHYCCD_ERROR)
        {
```

```
                printf("Close camera failed.\n");
                goto failure;
            }
        }

        //释放 SDK 资源
        ret = ReleaseQHYCCDResource();
        if(ret == QHYCCD_ERROR)
        {
            printf("Rlease SDK resource failed.\n");
            goto failure;
        }

        return 0;

failure:
        printf("Some fatal error happened.\n");
        return 1;
}
```

五、图像数据结构

通过 GetQHYCCDSingleFrame 和 GetQHYCCDLiveFrame 函数获取图像数据时，SDK 会将二维图像的像素数据按顺序读取出来依次存放到一维数组中，读取像素数据顺序为按照 Z 形读取顺序，即从左到右从上到下。

根据图像数据格式的不同，每个像素的数据结构也会发生变化，RAW8 或 MONO8 图像数据为一个 unsigned char 变量存储一个像素的数据；RAW16 或 MONO16 图像数据为两个 unsigned char 的变量存储一个像素的数据，此时需要考虑数据的高低位，低位在前，高位在后，像素值计算方式为高位数据乘以 256 加上低位数据；RGB24 图像数据为三个 unsigned char 数据存储一个像素的数据，彩色通道顺序为 BGR。当使用 unsigned char 类型一维数组存储时，具体如下：

RAW8 或 MONO8 格式数据：

ImgData[0]：第一行第一个像素数据，变量值即为像素值

ImgData[1]：第一行第二个像素数据，变量值即为像素值

后面以此类推。

RAW16 或 MONO16 格式数据：

ImgData[0]，ImgData[1]：第一行第一个像素，像素值为 ImgData[1]*256+ImgData[0]

ImgData[2]，ImgData[3]：第一行第二个像素，像素值为 ImgData[3]*256+ImgData[2]

后面以此类推。

RGB24 格式数据：

ImgData[0]，ImgData[1]，ImgData[2]：第一行第一个像素，变量值分别为 B、G、R 三个通道的像素值

ImgData[3]，ImgData[4]，ImgData[5]：第一行第二个像素，变量值分别为 B、G、R 三个通道的像素值

后面以此类推。