QHYCCD SDK说明文档

一、应用函数解析

所有二次开发中用得到的功能函数都声明在 qhyccd.h 头文件中，使用时只需要引用头文件即可。下面是应用函数的使用介绍：

1.1 基本操作函数

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。

示例代码：

```
int 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 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");
```

3.uint32_t OSXInitQHYCCDFirmware(char *path);

参数说明：

path	固件所在文件夹的存放位置，固件必须存放在 firmware 文件夹中，path 中记录的路径实际为 firmware 的存放路径；

函数说明：

用来加载固件的函数，暂时只有 Mac 上需要这个函数，Windows 上使用驱动，Linux 上使用 85-qhyccd.rules 文件。

示例代码：

```
int ret = QHYCCD_ERROR;
char path[] = "/usr/local";
ret = OSXInitQHYCCDFirmware(path);
if(ret == QHYCCD_SUCCESS){
    printf("Download firmware success!\n");
}else{
    printf("Download firmware fail!\n");
}
```

4.uint32_t ScanQHYCCD(void);

函数说明：

扫描已连接的 QHYCCD 设备，执行完成后会将扫描到的设备数量返回。

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");
}
```

5.uint32_t OpenQHYCCD(char *id);

参数说明：

id	GetQHYCCDId();返回的相机 ID;

函数说明：

会根据 GetQHYCCDId();返回的 ID 来打开相机，成功后返回相机的句柄。若句柄不为空则说明函数执行成功。之后所有对相机进行操作的函数都需要句柄作为参数。

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");
}
```

6.uint32_t CloseQHYCCD(qhyccd_handle *handle);

参数说明：

camhandle	OpenQHYCCD();返回的相机句柄;

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.");
```

7.uint32_t InitQHYCCD(qhyccd_handle *handle);

参数说明：

handle	OpenQHYCCD();返回的相机句柄;

函数说明：

初始化相机资源。成功返回 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");
}
```

1.2 相机的信息获取函数

用来获取相机的某项或某几项参数，可以根据获取的参数了解相机的某些信息，也可以将获取的参数可以做为后功能控制函数的参数，或作为判断的依据。

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

参数介绍：

index	相机结构体数组的下标，不能大于等于 ScanQHYCCD();的返回值；
id	一个类型的指针变量，用来承接函数返回的相机 ID；

函数说明：

获取已连接相机的 ID 号，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 QHY183C 是相机型号，后面 c915484fa76ea7552 是相机的序列号。每个相机都有其独有的序列号，即使是相同型号的不同相机，它们的序列号也是不同的。它的作用是区分相机，当做多相机测试时，这是很有必要的。

示例代码：

```
int i,ret;
char id[100][32] = {0};
for(i = 0;i < num;i++){
    ret = GetQHYCCDId(i,id[i]);
    if(ret == QHYCCD_SUCCESS){
        printf("Found connected camera,the id is %s\n",id[i]);
    }else{
        printf("some errors occered!(%d %d)\n",i,ret);
    }
}
```

2.uint32_t GetQHYCCDModel(char *id,char *model);

参数说明：

id	GetQHYCCDId();返回的相机 ID；
model	char 类型的数组，用来接收存储相机的型号；

函数说明：

获取相机的型号，如 QHY183C-c915484fa76ea7552 获取到的相机型号为 QHY183C。若函数成功执行，则返回 QHYCCD_SUCCESS。

示例代码：

```
char model[20];
ret = GetQHYCCDModel(id,model);
if(ret = QHYCCD_SUCCESS)
    printf("Camera model is %s.\n",model);
else
    printf("Get camera model fail.\n");
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
buf	用来存储固件版本信息的数组；

函数说明：

获取固件版本，只有在使用固件的 Linux 和 Mac 上才会用到这个函数，Windows 平台则不需要。可以根据获取到的固件版本判断当前使用的是否是最新的固件。若函数成功执行，则返回 QHYCCD_SUCCESS。

示例代码：

```
unsigned char buf[32];
ret = GetQHYCCDFWVersion(camhandle,buf);
if(ret = QHYCCD_SUCCESS)
    printf("year:%d month:%d day:%d\n",(buf[0] >> 4) + 0x10,buf[0]&~0xf0,buf[1]);
else
    printf("Get QHYCCD firmware version fail.\n");
```

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

参数说明：

year	接收 SDK 版本的年份；
month	接收 SDK 版本的月份；
day	接收 SDK 版本的日期；
subday	值为零，可忽略；

函数说明：

获取 SDK 的版本，即 SDK 的发布日期，所有平台都可以使用此函数，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,%d\n",year,month,day,subday);
else
    printf("Get QHYCCD SDK version fail.\n");
```

5.uint32_t GetQHYCCDType(qhyccd_handle *handle);

参数说明：

handle	OpenQHYCCD()返回的相机句柄

函数说明：

获取设备类型，如 DEVICETYPE_QHY183C（4045），若函数成功执行，则返回定义在 qhyccdcamdef.h 中的宏。

示例代码：

```
ret = GetQHYCCDType(camhandle);
if(ret != QHYCCD_ERROR)
    printf("Type:%d\n",ret);
else
    printf("Get QHYCCD Type fail.\n");
```

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

参数说明：

handle	OpenQHYCCD()返回的相机句柄；

controlId	代表相机某项功能参数的宏；
min	该参数允许设置的最小值；
max	改参数允许设置的最大值；
step	参数设置的步长；

函数说明：

获取某个相机参数的最大最小值及步长，可以根据这个函数获取的参数知道相机某项参数（如增益、偏置等）的设置范围及参数的设置步长。

示例代码：

```
double min,max,step;
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
if(ret == QHYCCD_SUCCESS)
{
    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″);
}
else
    printf(″Can’ t set gain\n″);
```

7.uint32_t GetQHYCCDHumidity(qhyccd_handle *handle,double *hd);

参数说明：

handle	OpenQHYCCD()返回的相机句柄；
hd	用来接收湿度信息的变量；

函数说明：

获取相机所处环境的湿度，暂时只有 A 系列和 IC16803 实现了这个功能。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
double hd;
ret = GetQHYCCDHumidity(camhandle,hd);
if(ret == QHYCCD_SUCCESS)
    printf(″HD:%lf\n″,hd);
else
    printf(″Get QHYCCD humidity fail.\n″);
```

8.uint32_t GetQHYCCDCameraStatus(qhyccd_handle *handle,uint8_t *buf);

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
buf	用来接收相机的运行状态；

函数说明：

获取相机的工作状态，包括闲置、等待、曝光和数据读取四个状态，暂时只有 A 系列相机实现了此函数。若函数执行成功，则返回 QHYCCD_SUCCESS。

buf[0] buf[1] buf[2] buf[3]

00 fe 81 74：闲置，相机不进行曝光操作和数据传输操作

01	fe	81	74：等待，相机曝光开始前的一段时间，很短
02	fe	81	74：曝光，开始时打开快门，结束后关闭快门
03	fe	81	74：数据读取

示例代码：

```
char buf[64];
ret = GetQHYCCDCameraStatus(camhandle,buf);
if(ret == QHYCCD_SUCCESS)
   printf("buf[0] = %x\n",buf[0]);
else
   printf("Get QHYCCD camera status error.\n");
```

9.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	OpenQHYCCD();返回的相机句柄；
chipw	镜片宽度；
chiph	镜片高度；
w	图像的宽度；
h	图像的高度；
pixelw	像素的宽度；
pixelh	像素的高度；
bpp	图像位深；

函数说明：

获取相机的片上信息，包括镜片的长度宽度、图像的长度宽度、像素的长度宽度和图像的位深。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);//获取相机信息
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");
}
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄;
startX	图像有效区域在水平方向的起始位置;
startY	图像有效区域在垂直方向的起始位置;
sizeX	图像有效区域的宽度;
sizeY	图像有效区域的高度;

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。

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");
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄;
startX	过扫区域在水平方向的起始位置;
startY	过扫有效区域在垂直方向的起始位置;
sizeX	过扫有效区域的宽度;
sizeY	过扫有效区域的高度;

函数说明：

有些 CCD 有过扫区域。这个函数将输出过扫区的 startx,starty sizex,sizey 参数。这个数据在原始图像中是物理上的。若函数执行成功，则返回 QHYCCD_SUCCESS。

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");
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄;

controlId	代表相机功能参数的宏，是定义在 qhyccdstruct.h 中的一个枚举类型；

这里列举几个常用的 ID：

```
CAM_COLOR,                //检查相机是否是彩色相机
CAM_BIN1X1MODE,           //检查相机是否具有 1X1bin 模式
CAM_BIN2X2MODE,           //检查相机是否具有 2X2bin 模式
CAM_BIN3X3MODE,           //检查相机是否具有 3X3bin 模式
CAM_BIN4X4MODE,           //检查相机是否具有 4X4bin 模式
CAM_MECHANICALSHUTTER,    //检查相机是否应用机械快门
CAM_GPS,                  //检查相机是否具有 GPS
CONTROL_COOLER            //检查相机是否是制冷型相机
CONTROL_CHANNELS,         //用于获取相机图像的通道数
CONTROL_CURTEMP,          //用于获取相机当前的温度
CONTROL_CURPWM,           //用于获取相机当前的制冷功率
CONTROL_MANULPWM,         //用于手动设置相机制冷功率
CONTROL_WBR,              //用于调节红光的白平衡
CONTROL_WBB,              //用于调节蓝光的白平衡
CONTROL_WBG,              //用于调节绿光的白平衡
CONTROL_GAIN,             //用于调节相机增益
CONTROL_OFFSET,           //用于设置相机偏置
CONTROL_EXPOSURE,         //用于设置相机的曝光时间
CONTROL_SPEED,            //用于设置 USB 的传输速度
CONTROL_TRANSFERBIT,      //用于获取相机图像位深
CONTROL_USBTRAFFIC,       //用于调节帧率
```

函数说明：

根据定义好的宏判断相机是否具有某项功能，若具有某项功能则返回 QHYCCD_SUCCESS，否则返回 QHYCCD_ERROR，若判断相机是否是彩色相机（CAM_COLOR），成功返回相机的 bayer 顺序，失败返回 QHYCCD_ERROR。相机的 bayer 顺序定义在 qhyccdstruct.h 文件中，定义如下：

```
enum BAYER_ID
{
    BAYER_GB = 1,
    BAYER_GR,
    BAYER_BG,
    BAYER_RG
};
```

示例代码：

1.检查相机是否是制冷型相机；

```
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_COOLER);
if(ret == QHYCCD_SUCCESS)
   printf("This camera is cooler camera.\n");
else
   printf("This camera is not cooler camera.\n");
```

2.检查相机是否是彩色相机；

```
ret = IsQHYCCDControlAvailable(camhandle,CAM_COLOR);
```

```
if(ret == BAYER_GB | ret == BAYER_GR | ret == BAYER_BG | ret == BAYER_RG)
    printf("This camera is color camera.\n");
else
    printf("This camera is not color camera.\n");
```

注：全部命令字定义及说明

其中一些命令字的作用仅仅是检查相机是否具有某项功能，一些命令字既可以用来检查相机是否具有某项功能，也可以设置相机的对应参数，剩下的一些命令字的作用是供 SDK 内部作为某些功能参数或状态的标志位而使用。

```
enum CONTROL_ID
{
    CONTROL_BRIGHTNESS = 0,                  //用于设置图像亮度
    CONTROL_CONTRAST,                        //用于设置图像对比
    CONTROL_WBR,                             //用于红光白平衡设置
    CONTROL_WBB,                             //用于设置蓝光白平衡
    CONTROL_WBG,                             //用于设置绿光白平衡
    CONTROL_GAMMA,                           //用于 Gamma 校正
    CONTROL_GAIN,                            //用于设置相机增益
    CONTROL_OFFSET,                          //用于设置相机 offset
    CONTROL_EXPOSURE,                        //用于设置曝光时间(us)
    CONTROL_SPEED,                           //用于设置相机的 USB 传输速度
    CONTROL_TRANSFERBIT,                     //用于设置或获取相机的图像位深
    CONTROL_CHANNELS,                        //用于设置或获取图像通道数
    CONTROL_USBTRAFFIC,                      //用于设置相机带宽
    CONTROL_ROWNOISERE,                      //行降噪
    CONTROL_CURTEMP,                         //用于获取相机当前的温度
    CONTROL_CURPWM,                          //用于获取相机当前的制冷功率
    CONTROL_MANULPWM,                        //用于手动设置制冷功率
    CONTROL_CFWPORT,                         //用于检查相机是否可连接滤镜轮
    CONTROL_COOLER,                          //用于检查是否是制冷型相机
    CONTROL_ST4PORT,                         //用于检查相机是否具有 ST4PORT
    CAM_COLOR,                               //用于检查是否是彩色相机
    CAM_BIN1X1MODE,                          //用于检查相机是否具有 1X1 bin 模式
    CAM_BIN2X2MODE,                          //用于检查相机是否具有 2X2 bin 模式
    CAM_BIN3X3MODE,                          //用于检查相机是否具有 3X3 bin 模式
    CAM_BIN4X4MODE,                          //检查相机是否具有 4X4 bin 模式
    CAM_MECHANICALSHUTTER,                   //检查相机是否具有机械快门
    CAM_TRIGER_INTERFACE,                    //用于检查相机是否具有外触发模式
    CAM_TECOVERPROTECT_INTERFACE,            //TEC 保护，限制制冷器功率
    CAM_SINGNALCLAMP_INTERFACE,              //用于检查相机是否具有信号灯
    CAM_FINETONE_INTERFACE,                  //用于检查相机是否具有 FINETONE 功能
    CAM_SHUTTERMOTORHEATING_INTERFACE,       //快门电机加热
    CAM_CALIBRATEFPN_INTERFACE,              //FPN 校正
    CAM_CHIPTEMPERATURESENSOR_INTERFACE,     //片上温度传感器
    CAM_USBREADOUTSLOWEST_INTERFACE,         //USB 以最低速读出数据
    CAM_8BITS,                              //检查相机的位深是否是 8bits
    CAM_16BITS,                             //检查相机的位深是否是 16bits
    CAM_GPS,                                //检查相机是否具有 GPS

    CAM_IGNOREOVERSCAN_INTERFACE,           //忽略 overscan 区
    QHYCCD_3A_AUTOBALANCE,                  //自动平衡
    QHYCCD_3A_AUTOEXPOSURE,                 //自动曝光
    QHYCCD_3A_AUTOFOCUS,                    //自动调焦
    CONTROL_AMPV,                           //用于检查相机是否具有 AMPV 功能
    CONTROL_VCAM,                           //虚拟相机开关
    CAM_VIEW_MODE,                          //视图模式

    CONTROL_CFWSLOTSNUM,                    //检查滤镜轮的槽数
    IS_EXPOSING_DONE,                       //检查相机是否已经曝光
    ScreenStretchB,                         //屏幕拉伸
    ScreenStretchW,                         //屏幕拉伸
    CONTROL_DDR,                            //检查相机是否具有 DDR 缓冲区
    CAM_LIGHT_PERFORMANCE_MODE,             //HGC/LGC 增益控制（已合并至增益控制中）

    CAM_QHY5II_GUIDE_MODE,                  //导星
    DDR_BUFFER_CAPACITY,                    //DDR 内容量
    DDR_BUFFER_READ_THRESHOLD               //DDR 缓冲区读阈值，超过这个阈值开始读 DDR
};
```

13. uint32_t GetQHYCCDParam(qhyccd_handle *handle, CONTROL_ID controlId);

参数说明：

handle	OpenQHYCCD();返回的相机句柄;
controlId	代表相机参数的宏，定义在 qhyccdstruct.h 中;

函数说明：

会根据 CONTROL_ID 对获取相机的功能参数的信息,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。

示例代码：

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

1.3 相机的功能控制函数

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
controlId	代表相机设置参数的宏，定义在 qhyccdstruct.h 中；
value	对应参数的值，参数不同，类型也不同；

函数说明：

根据定义好的宏对相机参数进行设置，若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

1.获取并设置曝光时间

```
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″);
```

2.设置相机带宽

```
ret = SetQHYCCDParam(camhandle,CONTROL_USBTRAFFIC,50);
if(ret == QHYCCD_SUCCESS)
   printf(“Set camera exposure time success.\n”);
else
   printf(“Set camera exposure time failed.\n”);
```

3.设置相机增益

```
ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,15);
if(ret == QHYCCD_SUCCESS)
   printf(“Set camera gain success.\n″);
else
   printf(“Set camera gain failed.\n”);
```

4.设置相机偏置

```
ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,140);
if(ret == QHYCCD_SUCCESS)
    printf(“Set camera gain success.\n”);
else
    printf(“Set camera gain failed.\n”);
```

5.设置相机的传输速度

```
ret = SetQHYCCDParam(camhandle,CONTROL_SPEED,1);
if(ret == QHYCCD_SUCCESS)
   printf(“Set camera transfer speed success.\n”);
else
   printf(“Set camera transfer speed failed.”);
```

6.温度控制

```
#define COOLER_ON    1
#define COOLER_OFF   2
```

```
#define COOLER_MANU 3
#define COOLER_AUTO 4

int   ret = QHYCCD_ERROR;
int   Flag_Cooler,Flag_Timer,Flag_Mode;
int   nowPWM,targetPWM;
float nowTemp,targetTemp;

ret = IsQHYCCDControlAvailable(camhandle,CONTROL_COOLER);
if(ret == QHYCCD_SUCCESS){
    printf( “Can operate this camera temperature control.\n″);
    Flag_Timer = 1;
    while(1){
        if(Flag_Cooler == COOLER_ON){
            if(Flag_Timer == 1){
                nowTemp = GetQHYCCDParam(camhandle,CONTROL_CURTEMP);
                nowPWM  = GetQHYCCDParam(camhandle,CONTROL_CURPWM);
                printf(″Now camera temperature is %.1f ° C,PWM
is %.1f%%.\n″,nowTemp,(float)nowPWM/255 * 100);
                Flag_Timer = Flag_Timer * -1;
                sleep(2);
            }else{
                if(Flag_Mode == COOLER_MANU){
                    ret = SetQHYCCDParam(camhandle,CONTROL_MANULPWM,targetPWM);
                    if(ret == QHYCCD_SUCCESS)
                        printf(″Set camera manu cooler success!\n″);
                    else
                        printf(″Set camera manu cooler failed!(%d)\n″,ret);
                }else if(Flag_Mode == COOLER_AUTO){
                    ret = SetQHYCCDParam(camhandle, CONTROL_COOLER, targetTemp);
                    if(ret == QHYCCD_SUCCESS)
                        printf(″Set camera auto cooler success!\n″);
                    else
                        printf(″Set camera auto cooler failed!(%d)\n″,ret);
                }
                Flag_Timer = Flag_Timer * -1;
                sleep(1);
            }
        }else if(Flag_Cooler == COOLER_OFF){
            ret = SetQHYCCDParam(camhandle, CONTROL_MANULPWM, 0);
            if(ret == QHYCCD_SUCCESS)
                printf(″Close camera cooler success!\n″);
                break;
            else
```

```
                printf("Close camera cooler failed!(%d)\n",ret);
        }else{
            printf("Cooler command error,please input right command.\n");
            Flag_Cooler = COOLER_ON;
        }
    }
}else
    printf("You can't set this camera input Auto_Cooler mode.\n");
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
targettemp	设定的目标温度；

函数说明：

控制相机制冷，和 SetQHYCCDParam(CONRTOL_COOLER)相同，成功执行则返回 QHYCCD_SUCCESS。若使用前不知道相机是否具有制冷功能，需要用 IsControlAvailable();函数进行判断。

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");
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
onoff	设置彩色模式的开关，是一个布尔类型的变量；

函数说明：

用来设置彩色相机的彩色模式的开和关，设置为 true 表示开启彩色模式，设置为 false 表示关闭彩色模式。只对彩色相机有效，在调用此函数之前需要先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");
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
wbin	水平方向上的 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。需要与SetQHYCCDResolution();函数配合使用。执行成功,则返回QHYCCD_SUCCESS。

示例代码：

详看SetQHYCCDResolution();函数的示例代码。

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
x	设置为0；
y	设置为0；
xsize	图像的宽度；
ysize	图像的高度；

函数说明：

用来设置相机的分辨率，需要与SetQHYCCDBinMode();配合使用。成功返回QHYCCD_SUCCESS。

示例代码：

```
ret = SetQHYCCDBinMode(camhandle,2,2);
if(ret = QHYCCD_SUCCESS){
    ret = SetQHYCCDResolution(camhandle,0,0,imagew/2,imageh/2);
    if(ret == QHYCCD_SUCCESS)
        printf("Set camera resolution success.\n");
    else
        printf("Set camera resolution fail.\n");
}else
    printf("Set camera bin mode fail.\n");
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
mode	相机的工作模式，0代表单帧模式，1代表连续模式；

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：

```
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”);
}
```

7.uint32_t GetQHYCCDMemLength(qhyccd_handle *handle);

参数说明：

handle	OpenQHYCCD();返回的相机句柄；

函数说明：

获取相机图像数据的内存长度，可以根据返回值为图像数据开辟空间。

示例代码：

```
int memlength = 0;
memlength = GetQHYCCDMemLength(camhandle);
if(memlength > 0)
    printf(“Get memory length success.\n”);
else
    printf(“Get memory length failed.\n”);
```

8.uint32_t ExpQHYCCDSingleFrame(qhyccd_handle *handle);

参数说明：

handle	OpenQHYCCD();返回的相机句柄;

函数说明：

开始曝光一帧图像，曝光时间由 SetQHYCCDParam(CONTROL_EXPOSURE)进行设置，单位为微妙。若函数执行成功，则返回 QHYCCD_SUCCESS。

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”);
```

9.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	OpenQHYCCD();返回的相机句柄;
w	图像宽度;
h	图像高度;
bpp	图像数据的位深;
channels	图像数据的通道数;
imgdata	用来接收图像数据;

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”);
```

10.uint32_t CancelQHYCCDExposingAndReadout(qhyccd_handle *handle);

参数说明：

handle	OpenQHYCCD();返回的相机句柄;

函数说明：

停止相机曝光并且停止数据读取。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，或相机输出数据且软件接收数据，否则软件或相机中的一个会卡死。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”);
```

11.uint32_t CancelQHYCCDExposing(qhyccd_handle *handle);

参数说明：

handle	OpenQHYCCD();返回的相机句柄;

函数说明：

停止相机曝光，但不停止相机数据输出。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”);
```

12.uint32_t BeginQHYCCDLive(qhyccd_handle *handle);

参数说明：

handle	OpenQHYCCD();返回的相机句柄;

函数说明：

开始连续模式曝光，曝光开始后会持续产生数据，上位机也应持续读出数据并显示。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”);
```

13.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	OpenQHYCCD();返回的相机句柄;
w	图像宽度;
h	图像高度;
bpp	图像数据的位深;
channels	图像数据的通道数;

ImgData	用来接收图像数据;

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”);
```

14. uint32_t StopQHYCCDLive(qhyccd_handle *handle);

参数说明:

handle	OpenQHYCCD();返回的相机句柄;

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");
```

15. void HistInfo192x130(qhyccd_handle *handle, uint32_t x, uint32_t y, uint8_t *InBuf, uint8_t *OutBuf);

参数说明:

handle	OpenQHYCCD();返回的相机句柄;
x	图像的实际宽度，imagew/bin;
y	图像的实际高度，imageh/bin;
InBuf	GetQHYCCDSingleFrame()获取到的图像数据;

函数说明:

根据读取的图像数据获取直方图信息。成功返回 QHYCCD_SUCCESS。

示例代码:

```
int ret = QHYCCD_ERROR;
ret = GetQHYCCDSingleFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
if(ret == QHYCCD_SUCCESS)
   HistInfo192x130(w,h,ImgData,outBuf);
else
    printf(“Get camera single frame failed.\n”);
```

Qt 实例:

```
QImage IplImageToQImage(const IplImage *iplImage){
    QImage *image = NULL;
    uchar *imgData;
    switch(iplImage->depth){
        case IPL_DEPTH_8U:
```

```
        {
            imgData=(uchar *)iplImage->imageData;
            if(iplImage->nChannels == 1)
                image = new QImage(imgData,iplImage->width,iplImage->height,
iplImage->widthStep, QImage::Format_Indexed8);
            else if(iplImage->nChannels == 3)
                image = new QImage(imgData,iplImage->width,iplImage->height,
iplImage->widthStep, QImage::Format_RGB888);
            else
                printf("IplImageToQImage: image format is not supported : depth=8U and
channels = %d",iplImage->nChannels);

        }
        break;
        default:
            printf("image format is not supported\n");
    }
    return image;
}
void displayHistogramImage(int x, int y, unsigned char *buf){
    IplImage *histImg = cvCreateImage(cvSize(192,130), IPL_DEPTH_8U, 3 );
    unsigned char *outBuf = (unsigned char*)malloc(35000000);
    if(outBuf){
        HistInfo192x130(x,y,buf,outBuf);
        histImg->imageData = (char*)outBuf;
        cvCvtColor(histImg,histImg,CV_BGR2RGB);
        QImage *histgramQImg = IplImageToQImage(histImg);

        managerMenu->ui->img_hist->setPixmap(QPixmap::fromImage(*histgramQImg));

        free(outBuf);
        cvReleaseImage(&histImg);
    }
}
```

16.double GetQHYCCDReadingProgress(qhyccd_handle *handle);

参数说明:

handle	OpenQHYCCD();返回的相机句柄;

函数说明:

获取图像数据的读取进度，暂时只有 QHY23 及 A 系列相机实现此函数，成功返回进度，失败则返回 QHYCCD_ERROR。需要在 ExpQHYCCDSingleFrame();函数前调用此函数，由于返回的进度只是估值，所以可能不会完全准确，甚至超过 100%。

示例代码:

```
double value;
value = GetQHYCCDReadingProgress(camhandle);
```

```
if(value >= 0)
    printf(“It’s %.1lf%%.\n”,value);
else
    printf(“Get QHYCCD read progress error.\n”);
```

17.uint32_t SetQHYCCDTrigerFunction(qhyccd_handle *handle,bool value);

参数说明：

handle	OpenQHYCCD()返回的相机句柄
value	布尔类型的变量，用来控制外触发是能与否，1：使能，0：不使能；

函数说明：

设置相机的外触发功能，外触发使能时，相机不会立即开始曝光而是等到外触发信号到了才开始。在使能外触发时，若在相机等待外触发的过程中，设置外触发不使能，就可以退出等待状态。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
ret = IsQHYCCDControlAvailable(camhandle,CAM_TRIGER_INTERFACE);
if(ret == QHYCCD_SUCCESS){
    ret = SetQHYCCDTrigerFunction(camhandle,true);
    if(ret == QHYCCD_SUCCESS)
        printf(“Open QHYCCD triger success.\n”);
    else
        printf(“Open QHYCCD triger fail.\n”);
}else
    printf(“Can’t set triger.\n”);
```

18.uint32_T IsQHYCCDCFWPlugged(qhyccd_handle *handle);

参数说明：

handle	OpenQHYCCD()返回的相机句柄

函数说明：

滤镜轮是否已连接，只有 QHY5IIICOOL 系列和 MINICAM5F_M 实现了此函数，必须确保滤镜轮已通电才能返回 QHYCCD_SUCCESS，否则视为未连接，返回 QHYCCD_ERROR;

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”);
```

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

参数说明：

handle	OpenQHYCCD()返回的相机句柄
status	用来接收滤镜轮的当前位置

函数说明：

获取滤镜轮状态，第几孔，范围是 0 到滤镜轮的孔数减 1，若滤镜轮实际孔数为八，则实际对应的为 0~7;

示例代码：

```
char status[64];
char dst;
ret = GetQHYCCDCFWStatus(camhandle,status);
if(ret = QHYCCD_SUCCESS){
   if(dst == status[0])
      printf(“CFW has moved.\n”);
   else
      printf(“CFW is moving.\n”);
}else
   printf(“Get QHYCCD CFW status error.\n”);
```

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

参数说明:

handle	OpenQHYCCD()返回的相机句柄
order	设置滤镜轮的目标位置
length	order 的字符长度

函数说明:

控制滤镜轮转动，order 是目标孔，为实际目标孔数减一

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”);
```

21.void SetQHYCCDLogLevel(uint8_t logLevel);

参数说明:

loglevel	设置日志信息的输出等级;

函数说明:

输出日志信息到终端或控制台，根据参数的设置，可以输出不同的日志信息，0：LOG_DEBUG，1：LOG_TRACE。

示例代码:

```
SetQHYCCDLevel(0);
SetQHYCCDLevel(1);
```

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

参数说明:

handle	OpenQHYCCD();返回的相机句柄;
i	用来控制 VCOX 频率的参数，范围是 0~4095

函数说明:

用来控制 GPS 相机的 VCOX 频率，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");
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
i	用来设置 LED 灯使能的参数，0：不使能，1：使能；

函数说明：

用来控制校准 LED 灯使能的函数，若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int i = 1;
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");
```

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

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
i	用来设置相机主从模式的参数，0：主模式，1：从模式；

函数说明：

用来控制 GPS 相机的主从模式，若函数执行成功，则返回 QHYCCD_SUCCESS。当处于从模式时，使用 SetQHYCCDGPSSlaveModeParameter(qhyccd_handle *handle,uint32_t target_sec,uint32_t target_us,uint32_t deltaT_sec,uint32_t deltaT_us,uint32_t expTime)设置参数。target_sec 是 QHYCCD 定义的“JS”。它指的是一段时间。

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");
```

25.void SetQHYCCDGPSPOSA(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	设置 LED 脉冲宽度；

函数说明：

设置 LED 脉冲位置，用于快门曝光。当你改变了曝光时间，你必须设置这个位置。测量电路将使用这个位置作为快门启动时间。

示例代码：

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

26.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	设置 LED 脉冲宽度；

函数说明：

设置 LED 脉冲位置，用于快门曝光。当你改变了曝光时间，你必须设置这个位置。测量电路将使用这个位置作为快门结束时间。

示例代码：

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

补充：图像的数据结构头。

摄像机记录下 GPS 信息并插入每个帧的头部，可以通过 API 来启用和禁用：

启用：ret=SetQHYCCDParam(g_hCam,CAM_GPS,1);

禁用：ret=SetQHYCCDParam(g_hCam,CAM_GPS,0);

27.void Bits16ToBits8(qhyccd_handle *handle, uint8_t *InputData16, uint8_t *OutputData8, uint32_t imageX, uint32_t imageY, uint16_t B, uint16_t W);

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
InputData16	输入的 16 位图像数据；
OutputData8	输出的 8 位图像数据；
imageX	图像的宽度；
imageY	图像的高度；
B	用来设置灰度拉伸的参数；
W	用来设置灰度拉伸的参数；

函数说明：

16 位数据转换位 8 位，同时进行灰度拉伸。

示例代码：

```
int imageX = 1280,imageY = 960,B = 20000,W = 30000;
Bits16toBits8(camhandle,InputData,OutputData,imageX,imageY,B,W);
```

28.uint32_t SetQHYCCDFocusSetting(qhyccd_handle *h,uint32_t focusCenterX, uint32_t focusCenterY);

参数说明：

handle	OpenQHYCCD();返回的相机句柄；
focusCenterX	焦点中心的 X 坐标；
focusCenterY	焦点中心的 Y 坐标；

函数说明：

用于设置调焦模式,不同相机用的 BIN 和 ROI 不同，设置方式也不同。若函数执行成功，则返回 QHYCCD_SUCCESS。

示例代码：

```
int x = 640,y = 480;
ret = SetQHYCCDFocusSetting(camhandle,x,y);
if(ret == QHYCCD_SUCCESS)
   printf("Set QHYCCD focus setting success.\n");
else
   printf("Set QHYCCD focus setting fail.\n");
```

29.uint32_t SetQHYCCDFineTone(qhyccd_handle *handle,uint8_t setshporshd,uint8_t shdloc,uint8_t shploc,uint8_t shwidth);

函数说明：

对应 QHY9 和 QHY11，用来优化 CCD 的驱动时序，可以进一步优化 CCD 的读出噪声。由于这个函数比较复杂，若有需要请联系我们的软件工程师。

30.uint32_t DownloadFX3FirmWare(uint16_t vid,uint16_t pid,char *imgpath);

参数说明：

vid	相机的 VID；
pid	相机的 PID；
imgpath	固件的存放位置；

函数说明：

为相机下载固件。若函数成功执行，则返回 QHYCCD_SUCCESS。

示例代码：

```
char path[] = "/usr/local/lib";
ret = DownloadFX3FirmWare(0x1618,0x183,path);
if(ret == QHYCCD_SUCCESS)
   printf("Download firmware success.\n");
else
   printf("Download firmware fail.\n");
```

1.4 示例程序

1.单帧模式

```
SingleFrameSample:
#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;

    ret = InitQHYCCDResource();
    if(ret == QHYCCD_SUCCESS)
    {
        printf("Init SDK success!\n");
    }
    else
    {
        goto failure;
    }

    num = ScanQHYCCD();
    if(num > 0)
    {
        printf("Yes!Found QHYCCD,the num is %d \n",num);
    }
    else
    {
        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;
            break;
        }
    }

    if(found == 1)
    {
        camhandle = OpenQHYCCD(id);
        if(camhandle != NULL)
        {
            printf("Open QHYCCD success!\n");
        }
        else
        {
            printf("Open QHYCCD fail \n");
            goto failure;
        }

        ret = SetQHYCCDStreamMode(camhandle,0);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("SetQHYCCDStreamMode success!\n");
        }
        else
        {
            printf("SetQHYCCDStreamMode code:%d\n",ret);
            goto failure;
        }

        ret = InitQHYCCD(camhandle);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Init QHYCCD success!\n");
        }
        else
        {
            printf("Init QHYCCD fail code:%d\n",ret);
            goto failure;
        }

        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;
        }
        ret = IsQHYCCDControlAvailable(camhandle,CAM_COLOR);
        if(ret == BAYER_GB || ret == BAYER_GR || ret == BAYER_BG || ret == BAYER_RG)
        {
            printf("This is a Color Cam\n");
            SetQHYCCDDebayerOnOff(camhandle,true);
            SetQHYCCDParam(camhandle,CONTROL_WBR,20);//设置相机的红光白平衡
            SetQHYCCDParam(camhandle,CONTROL_WBG,20);//设置相机的绿光白平衡
            SetQHYCCDParam(camhandle,CONTROL_WBB,20);//设置相机的蓝光白平衡
        }

        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_USBTRAFFIC);        if(ret ==
QHYCCD_SUCCESS)
        {
            ret = SetQHYCCDParam(camhandle,CONTROL_USBTRAFFIC,30);            if(ret !=
QHYCCD_SUCCESS)
            {
                printf("SetQHYCCDParam CONTROL_USBTRAFFIC failed\n");
                getchar();
                return 1;
            }
        }

        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
        if(ret == QHYCCD_SUCCESS)
        {
            ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,30);
            if(ret != QHYCCD_SUCCESS)
            {
                printf("SetQHYCCDParam CONTROL_GAIN failed\n");
                getchar();
                return 1;
```

```
    }
}

ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
if(ret == QHYCCD_SUCCESS)
{
    ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,140);
    if(ret != QHYCCD_SUCCESS)
    {
        printf("SetQHYCCDParam CONTROL_GAIN failed\n");
        getchar();
        return 1;
    }
}

ret = SetQHYCCDParam(camhandle,CONTROL_EXPOSURE,2000000);
if(ret != QHYCCD_SUCCESS)
{
    printf("SetQHYCCDParam CONTROL_EXPOSURE failed\n");
    getchar();
    return 1;
}

ret = SetQHYCCDResolution(camhandle,0,0,w,h);
if(ret == QHYCCD_SUCCESS)
{
    printf("SetQHYCCDResolution success!\n");
}
else
{
    printf("SetQHYCCDResolution fail\n");
    goto failure;
}

ret = SetQHYCCDBinMode(camhandle,cambinx,cambiny);
if(ret == QHYCCD_SUCCESS)
{
    printf("SetQHYCCDBinMode success!\n");
}
else
{
    printf("SetQHYCCDBinMode fail\n");
    goto failure;
}
```

```
ret = IsQHYCCDControlAvailable(camhandle,CONTROL_TRANSFERBIT);
 if(ret == QHYCCD_SUCCESS)
 {
     ret = SetQHYCCDBitsMode(camhandle,16);
     if(ret != QHYCCD_SUCCESS)
     {
         printf("SetQHYCCDParam CONTROL_GAIN failed\n");
         getchar();
         return 1;
     }
 }

 ret = ExpQHYCCDSingleFrame(camhandle);
 if( ret != QHYCCD_ERROR )
 {
     printf("ExpQHYCCDSingleFrame success!\n");
     if( ret != QHYCCD_READ_DIRECTLY )
     {
         sleep(1);
     }
 }
 else
 {
     printf("ExpQHYCCDSingleFrame fail\n");
     goto failure;
 }

 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;
 }

 ret = GetQHYCCDSingleFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
 if(ret == QHYCCD_SUCCESS)
 {
     printf("GetQHYCCDSingleFrame succeess! \n");
```

```
         //show the image

     }
     else
     {
         printf(″GetQHYCCDSingleFrame fail:%d\n″,ret);
     }

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

 if(camhandle)
 {
     ret = CancelQHYCCDExposingAndReadout(camhandle);
     if(ret == QHYCCD_SUCCESS)
     {
         printf(″CancelQHYCCDExposingAndReadout success!\n″);
     }
     else
     {
         printf(″CancelQHYCCDExposingAndReadout fail\n″);
         goto failure;
     }

     ret = CloseQHYCCD(camhandle);
     if(ret == QHYCCD_SUCCESS)
     {
         printf(″Close QHYCCD success!\n″);
     }
     else
     {
         goto failure;
     }
 }

ret = ReleaseQHYCCDResource();
 if(ret == QHYCCD_SUCCESS)
 {
     printf(″Rlease SDK Resource  success!\n″);
```

```
    }
    else
    {
        goto failure;
    }

    return 0;

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

2.连续模式

LiveFrameSample:

```
#include <stdio.h>
#include <time.h>
#include "qhyccd.h"
#include "highgui.h"

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

    qhyccd_handle *camhandle;

    ret = InitQHYCCDResource();
    if(ret == QHYCCD_SUCCESS)
    {
        printf("Init SDK success!\n");
    }
    else
    {
        goto failure;
    }
    num = ScanQHYCCD();
    if(num > 0)
    {
        printf("Found %d QHYCCD device.\n",num);
    }
    else
    {
        printf("Not Found QHYCCD device,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 QHYCCD device.(id:%s)\n",id);
            found = 1;
            break;
        }
    }

    if(found == 1)
    {
        camhandle = OpenQHYCCD(id);
        if(camhandle != NULL)
        {
            printf("Open QHYCCD device success!\n");
        }
        else
        {
            printf("Open QHYCCD device failed!(%d)\n",ret);
            goto failure;
        }
        ret = SetQHYCCDStreamMode(camhandle,1);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Set QHYCCD device stream mode success!\n");
        }
        else
        {
            printf("Set QHYCCD device stream mode failed!(%d)\n",ret);
            goto failure;
        }

        ret = InitQHYCCD(camhandle);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Init QHYCCD device success!\n");
        }
        else
        {
            printf("Init QHYCCD device failed!(%d)\n",ret);
            goto failure;
        }

        ret = GetQHYCCDChipInfo(camhandle,&chipw,&chiph,&w,&h,&pixelw,&pixelh,&bpp);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Get QHYCCD ChipInfo success!\n");
            printf("CCD/CMOS chip information  :\n");
            printf("CCD/CMOS chip width        :%3f mm\n",chipw);
            printf("CCD/CMOS chip height       :%3f mm\n",chiph);
            printf("CCD/CMOS chip pixel width  :%3f um\n",pixelw);
            printf("CCD/CMOS chip pixel height :%3f um\n",pixelh);
            printf("CCD/CMOS chip width        :%d\n",w);
            printf("CCD/CMOS chip height       :%d\n",h);
            printf("CCD/CMOS chip depth        :%d\n",bpp);
        }
        else
        {
            printf("Get QHYCCD ChipInfo failed!(%d)\n",ret);
            goto failure;
        }

        ret = IsQHYCCDControlAvailable(camhandle,CAM_COLOR);
        if(ret == BAYER_GB || ret == BAYER_GR || ret == BAYER_BG || ret == BAYER_RG)
        {
            printf("This QHYCCD device is a color camera!\n");
            SetQHYCCDDebayerOnOff(camhandle,true);
            SetQHYCCDParam(camhandle,CONTROL_WBR,64);//set camera param by definition
            SetQHYCCDParam(camhandle,CONTROL_WBG,64);
            SetQHYCCDParam(camhandle,CONTROL_WBB,64);
        }else{
            printf("This QHYCCD device is not a color camera!\n");
        }

        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_DDR);
        if(ret == QHYCCD_SUCCESS){
            printf("This QHYCCD device has DDR!\n");
            ret = SetQHYCCDParam(camhandle,CONTROL_DDR,true);
            if(ret == QHYCCD_SUCCESS){
               printf("Open QHYCCD device DDR success!\n");
            }else{
               printf("Open QHYCCD device DDR failed!(%d)",ret);
            }
        }else{
            printf("This QHYCCD device doesn’t have DDR!\n");
        }

        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_TRANSFERBIT);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Can set this QHYCCD device transfer bits!\n");
            ret = SetQHYCCDBitsMode(camhandle,8);
            if(ret == QHYCCD_SUCCESS)
            {
                printf("Set QHYCCD device transfer bits success!\n");
            }else{
               printf("Set QHYCCD device transfer bits failed!(%d)\n",ret);
            }
        }else{
               printf("Can't set this QHYCCD device transfer bits!\n");
        }

        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Can set this QHYCCD device offset.\n");
            ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,50);
            if(ret == QHYCCD_SUCCESS)
            {
                printf("Set QHYCCD device offset success!\n");
            }else{
               printf("Set QHYCCD device offset failed!(%d)\n",ret);
            }
        }else{
            printf("Can't set this QHYCCD device offset!\n");
        }

        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Can set this QHYCCD device gain.");
            ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,50);
            if(ret == QHYCCD_SUCCESS){
               printf("Set QHYCCD device gain success!\n");
            }else{
                printf("Set QHYCCD device gain failed!(%d)\n",ret);
            }
        }else{
            printf("Can't set this QHYCCD device gain.");
        }

        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_USBTRAFFIC);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Can set this QHYCCD device USBTraffic!\n");
            ret = SetQHYCCDParam(camhandle,CONTROL_USBTRAFFIC,60);
            if(ret == QHYCCD_SUCCESS)
            {
                printf("Set QHYCCD device USBTraffic success!\n");
            }else{
                printf("Set QHYCCD device USBTraffic failed!(%d)\n",ret);
                goto failure;
            }
        }else{
            printf("Can't set this QHYCCD device USBTraffic!\n");
        }

        int exp_time = 0;
        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_EXPOSURE);
        if(ret == QHYCCD_SUCCESS){
            printf("Can set this QHYCCD device exposure time.\n");
            exp_time = GetQHYCCDParam(camhandle,CONTROL_EXPOSURE);
            if(exp_time > 0)
                printf("QHYCCD device exposure time is %3d ms.\n",exp_time/1000);
            else
                printf("Get QHYCCD device exposure time failed!(%d)\n",ret);

            ret = SetQHYCCDParam(camhandle,CONTROL_EXPOSURE,20*1000);
            if(ret == QHYCCD_SUCCESS){
                printf("Set QHYCCD device exposure time success!\n");
                exp_time = GetQHYCCDParam(camhandle,CONTROL_EXPOSURE);
                if(exp_time > 0)
                    printf("QHYCCD device exposure time is %3d ms.\n",exp_time/1000);
                else
                    printf("Get QHYCCD device exposure time failed!(%d)",ret);
            }else{
                printf("Set QHYCCD device exposure time failed!(%d)\n",ret);
            }
        }

        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_SPEED);
        if(ret == QHYCCD_SUCCESS){
            printf("Can set this QHYCCD device speed!\n");
                ret = SetQHYCCDParam(camhandle,CONTROL_SPEED,2);
                if(ret == QHYCCD_SUCCESS)
                {
                    printf("Set QHYCCD device speed succeed!\n");
               }else{
                    printf("Set QHYCCD device speed failed!(%d)\n",ret);
                    goto failure;
                }
            }else{
                printf(″Can’t set this QHYCCD device speed!\n″);
            }

            ret = IsQHYCCDControlAvailable(camhandle,CAM_BIN1X1MODE);
            if(ret == QHYCCD_SUCCESS){
                printf(″Can set this camera 1X1 bin mode.\n″);
            ret = SetQHYCCDBinMode(camhandle,1,1);
            if(ret == QHYCCD_SUCCESS){
                printf(″Set camera 1X1 bin mode success!\n″);
                ret = SetQHYCCDResolution(camhandle,0,0,w,h);
                if(ret == QHYCCD_SUCCESS)
                {
                    printf(″Set camera resolution success!\n″);
                }
                else
                {
                    printf(″Set camera resolution failed!(%d)\n″,ret);
                    goto failure;
                }
            }else{
                printf(″Set camera 1X1 bin mode failed!(%d)″,ret);
            }
        }

        int length = GetQHYCCDMemLength(camhandle);
        if(length > 0)
        {
            printf(″Get camrea memory length success!\n″);
            ImgData = (unsigned char *)malloc(length*2);
            memset(ImgData,0,length);
        }
        else
        {
            printf(″Get camera memory length failed!(%d)\n″,ret);
            goto failure;
        }

        int t_start,t_end;
        t_start = time(NULL);
        int fps = 0,t_num = 0;

        ret = BeginQHYCCDLive(camhandle);
```

```
if(ret == QHYCCD_SUCCESS)
{
    printf("BeginQHYCCDLive success!\n");
    cvNamedWindow("show",0);

    while(ret == QHYCCD_SUCCESS)
    {
        while(ret_live == QHYCCD_ERROR){
          ret_live = GetQHYCCDLiveFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
        }
        IplImage *image = cvCreateImage(cvSize(w,h),bpp,channels);
        image->imageData = (char *)ImgData;

        cvShowImage("show",image);
        cvWaitKey(5);
        ret_live = QHYCCD_ERROR;
        fps++;

        t_end = time(NULL);
        if(t_end - t_start >= 1){
            t_num ++;
            if(t_num % 5 == 0){
                printf("Time pass:%3d | Frame rate:%5.1f\n",t_num,(float)fps/5);
                fps = 0;
            }else
                printf("Time pass:%3d | \n",t_num);
            t_start = time(NULL);
        }

        if(t_num >= 120)
        {
            //break;
            ret = QHYCCD_ERROR;
        }
    }
}
else
{
    printf("BeginQHYCCDLive failed\n");
    goto failure;
}

if(ImgData != NULL){
    delete(ImgData);
```

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

    if(camhandle)
    {
        ret = StopQHYCCDLive(camhandle);
        if(ret == QHYCCD_SUCCESS){
               printf("Stop QHYCCD live success!\n");
        }

        ret = CloseQHYCCD(camhandle);
        if(ret == QHYCCD_SUCCESS)
        {
            printf("Close QHYCCD success!\n");
        }
        else
        {
            goto failure;
        }
    }

    ret = ReleaseQHYCCDResource();
    if(ret == QHYCCD_SUCCESS)
    {
        printf("Rlease SDK Resource  success!\n");
    }
    else
    {
        goto failure;
    }
    return 0;

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

3.相机制冷控制

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include "qhyccd.h"
#include "opencv2/highgui/highgui.hpp"

#define COOLER_ON   1
#define COOLER_OFF  2
#define COOLER_MANU 3
#define COOLER_AUTO 4

qhyccd_handle *camhandle = NULL;
int Flag_Cooler,Flag_Timer,Flag_Mode;
int targetPWM;
float targetTemp = -5;

void *Cooler_Control(void *)
{
        int ret = QHYCCD_ERROR;
        float nowTemp;
        int nowPWM;
        Flag_Timer = 1;

        ret = IsQHYCCDControlAvailable(camhandle,CONTROL_COOLER);
        if(ret == QHYCCD_SUCCESS){
            printf("You can set this camera input Auto_Cooler mode.\n");
            while(1){
                if(Flag_Cooler == COOLER_ON){
                    if(Flag_Timer == 1){
                        nowTemp = GetQHYCCDParam(camhandle,CONTROL_CURTEMP);
                        nowPWM  = GetQHYCCDParam(camhandle,CONTROL_CURPWM);
                        printf("Now camera temperature is %.1f ° C,PWM
is %.1f%%.\n",nowTemp,(float)nowPWM/255 * 100);
                        Flag_Timer = Flag_Timer * -1;
                        sleep(2);
                    }else{
                        if(Flag_Mode == COOLER_MANU){
                            ret = SetQHYCCDParam(camhandle,CONTROL_MANULPWM,targetPWM);
                            if(ret == QHYCCD_SUCCESS){
                                printf("Set camera manu cooler success!\n");
                            }else{
                                printf("Set camera manu cooler failed!(%d)\n",ret);
                            }
                        }else if(Flag_Mode == COOLER_AUTO){
                            ret = SetQHYCCDParam(camhandle, CONTROL_COOLER, targetTemp);
                            if(ret == QHYCCD_SUCCESS){
                                printf("Set camera auto cooler success!\n");
                            }else{
                                printf("Set camera auto cooler failed!(%d)\n",ret);
                            }
                        }
                        Flag_Timer = Flag_Timer * -1;
                        sleep(1);
                    }
                }else if(Flag_Cooler == COOLER_OFF){
                    ret = SetQHYCCDParam(camhandle, CONTROL_MANULPWM, 0);
                    if(ret == QHYCCD_SUCCESS){
                        printf("Close camera cooler success!\n");
                        break;
                    }else{
                        printf("Close camera cooler failed!(%d)\n",ret);
                    }
                }else{
                    printf("Cooler command error,please input right command.\n");
                    Flag_Cooler = COOLER_ON;
                }
            }
        }else{
            printf("You can’t set this camera input Auto_Cooler mode.\n");
        }
    pthread_exit(0);
}

int main(int argc,char *argv[])
{
    int num = 0;
    int ret = QHYCCD_ERROR;
    int found = 0;
    int cambinx = 1,cambiny = 1;
    unsigned char *ImgData;
    IplImage *image;
    unsigned int w,h,bpp,channels = 0;

    char id[32];
```

```
pthread_t tid_cooler;
pthread_t tid_getdata;

ret = InitQHYCCDResource();
if(ret == QHYCCD_SUCCESS)
{
    printf("Init SDK success!\n");
}
else
{
    goto failure;
}

num = ScanQHYCCD();
if(num > 0)
{
    printf("Yes!Found QHYCCD,the num is %d \n",num);
}
else
{
    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 QHYCCD success!\n");
    }
    else
    {
```

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



ret = SetQHYCCDStreamMode(camhandle,0);
if(ret == QHYCCD_SUCCESS)
{
    printf("SetQHYCCDStreamMode success!\n");
}
else
{
    printf("SetQHYCCDStreamMode code:%d\n",ret);
    goto failure;
}

ret = InitQHYCCD(camhandle);
if(ret == QHYCCD_SUCCESS)
{
    printf("Init QHYCCD success!\n");
}
else
{
    printf("Init QHYCCD fail code:%d\n",ret);
    goto failure;
}

Flag_Cooler = COOLER_ON;
Flag_Mode   = COOLER_AUTO;
pthread_create(&tid_cooler,NULL,Cooler_Control,NULL);//开始相机制冷
sleep(80);//等待相机温度稳定

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");
    goto failure;
}

ret = IsQHYCCDControlAvailable(camhandle,CAM_COLOR);
if(ret == BAYER_GB || ret == BAYER_GR || ret == BAYER_BG || ret == BAYER_RG)
{
    printf("This is a Color Camera\n");
    SetQHYCCDDebayerOnOff(camhandle,true);
    SetQHYCCDParam(camhandle,CONTROL_WBR,64);//set camera param by definition
    SetQHYCCDParam(camhandle,CONTROL_WBG,64);
    SetQHYCCDParam(camhandle,CONTROL_WBB,64);
}

ret = IsQHYCCDControlAvailable(camhandle,CONTROL_USBTRAFFIC);
if(ret == QHYCCD_SUCCESS)
{
    ret = SetQHYCCDParam(camhandle,CONTROL_USBTRAFFIC,50);
    if(ret != QHYCCD_SUCCESS)
    {
        printf("SetQHYCCDParam CONTROL_USBTRAFFIC failed\n");
        getchar();
        return 1;
    }
}

ret = IsQHYCCDControlAvailable(camhandle,CONTROL_GAIN);
if(ret == QHYCCD_SUCCESS)
{
    ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,6);
    if(ret != QHYCCD_SUCCESS)
    {
        printf("SetQHYCCDParam CONTROL_GAIN failed!\n");
        getchar();
        return 1;
    }
}

ret = IsQHYCCDControlAvailable(camhandle,CONTROL_OFFSET);
```

```
if(ret == QHYCCD_SUCCESS)
{
    ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,150);
    if(ret != QHYCCD_SUCCESS)
    {
        printf("SetQHYCCDParam CONTROL_GAIN failed!\n");
        getchar();
        return 1;
    }
}

ret = SetQHYCCDParam(camhandle,CONTROL_EXPOSURE,1*1000000);
if(ret != QHYCCD_SUCCESS)
{
    printf("SetQHYCCDParam CONTROL_EXPOSURE failed!\n");
    getchar();
    return 1;
}

ret = SetQHYCCDParam(camhandle,CONTROL_SPEED,1);
if(ret == QHYCCD_SUCCESS)
{
       printf("SetQHYCCDParam CONTROL_SPEED succeed!\n");
}

ret = SetQHYCCDResolution(camhandle,0,0,w,h);//设置相机分辨率
if(ret == QHYCCD_SUCCESS)
{
    printf("SetQHYCCDResolution success!\n");
}
else
{
    printf("SetQHYCCDResolution failed!\n");
    goto failure;
}

ret = SetQHYCCDBinMode(camhandle,cambinx,cambiny);//设置相机输出图像数据的模式
if(ret == QHYCCD_SUCCESS)
{
    printf("SetQHYCCDBinMode success!\n");
}
else
{
    printf("SetQHYCCDBinMode failed!\n");
    goto failure;
}

uint32_t length = GetQHYCCDMemLength(camhandle);//获取相机内存长度
if(length > 0)
{
    ImgData = (unsigned char *)malloc(length*2);
    memset(ImgData,0,length);
    printf("QHYCCD | SingleFrameSample | camera length = %d\n",length);
}
else
{
    printf("Get the min memory space length failure\n");
    goto failure;
}

ret = ExpQHYCCDSingleFrame(camhandle);//开始曝光一帧图像
if(ret != QHYCCD_ERROR )
{
    printf("ExpQHYCCDSingleFrame success!\n");
    if(ret != QHYCCD_READ_DIRECTLY)
    {
       // sleep(1);
    }
}
else
{
    printf("ExpQHYCCDSingleFrame failed!\n");
    goto failure;
}

ret = GetQHYCCDSingleFrame(camhandle,&w,&h,&bpp,&channels,ImgData);
if(ret == QHYCCD_SUCCESS){
    printf("GetQHYCCDSingleFrame succeess!\n");
    image = cvCreateImage(cvSize(w,h),bpp,channels);
    image->imageData = (char *)ImgData;

    cvNamedWindow("qhyccd",0);
    cvShowImage("qhyccd",image);

    cvWaitKey(0);

    cvDestroyWindow("qhyccd");
    cvReleaseImage(&image);
    }else
        printf("GetQHYCCDSingleFrame fail:%d\n",ret);

    if(ImgData != NULL){
        printf("QHYCCD | SingleFrameSample.CPP | delete ImgData\n");
        delete(ImgData);
    }
}
else
{
    printf("The camera is not QHYCCD or other error \n");
    goto failure;
}

if(camhandle)
{
    ret = CancelQHYCCDExposingAndReadout(camhandle);//停止相机曝光和数据读取
    if(ret == QHYCCD_SUCCESS)
    {
        printf("CancelQHYCCDExposingAndReadout success!\n");
    }
    else
    {
        printf("CancelQHYCCDExposingAndReadout fail\n");
        goto failure;
    }

           Flag_Cooler = COOLER_OFF;
    //usleep(1);
    pthread_join(tid_cooler,0);

    ret = CloseQHYCCD(camhandle);//关闭相机
    if(ret == QHYCCD_SUCCESS)
    {
        printf("Close QHYCCD success!\n");
    }
    else
    {
        goto failure;
    }
}

ret = ReleaseQHYCCDResource();//释放相机资源
if(ret == QHYCCD_SUCCESS)
```

```
    {
        printf("Rlease SDK Resource  success!\n");
    }
    else
    {
        goto failure;
    }
        printf("QHYCCD | SingleFrameSample.cpp | end\n");
    return 0;

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

二、底层协议

底层协议：http://qhyccd.com/bbs/index.php?board=24.0
libusb 官网：http://libusb.info/

2.1 函数说明

Linux&Mac：

Linux 和 Mac 上通过 libusb 库使用底层协议，下面对常用的函数进行简单地介绍：

```
1.int LIBUSB_CALL libusb_init(libusb_context **ctx);
```

初始化函数，用来初始化 libusb-1.0 库，必须首先调用，参数一般为 NULL；

```
2.libusb_device_handle * LIBUSB_CALL libusb_open_device_with_vid_pid(
libusb_context *ctx,
uint16_t vendor_id,
uint16_t product_id);
```

用来打开设备，成功执行后可以获得 USB 设备的句柄，ctx 设置成 NULL 就行，vendor_id 和 product_id 分别对应 USB 设备的 VID 和 PID；

```
3.int LIBUSB_CALL libusb_control_transfer(
libusb_device_handle *dev_handle,
uint8_t request_type,
uint8_t bRequest,
uint16_t wValue,
uint16_t wIndex,
unsigned char *data,
uint16_t wLength,
unsigned int timeout);
```

控制传输函数，用来发送命令或从相机读取数据，dev_handle 是设备句柄，request_type 和 bRequest 用来设置数据传输方向，
发送命令给相机：0x40,0xD1
从相机读取数据：0xc0,0xD2

data 是要发送的命令或要读取的数据，wLength 是 data 的大小，timeout 是超出时间，设置为零即可，其他参数相机没用到，也设置为零。

```
4.int LIBUSB_CALL libusb_bulk_transfer(
libusb_device_handle *dev_handle,
unsigned char endpoint,
unsigned char *data,
int length,
int *actual_length,
unsigned int timeout);
```

块传输函数，用来读取相机里的图像数据，dev_handle 是设备句柄，endpoint 是端点号，可以用程序打印出来，data 用来接收数据，length 是 data 的大小，actual_length 定义一个同类型的指针即可，timeout 设置为零。

```
5.int LIBUSB_CALL libusb_kernel_driver_active(
libusb_device_handle *dev_handle,
int interface_number);
```

判断是否存在设备驱动，dev_handle 设备句柄，interface_number 设置为零。

```
6.int LIBUSB_CALL libusb_detach_kernel_driver(
libusb_device_handle *dev_handle,
int interface_number);
```

移除设备驱动，dev_handle 设备句柄，interface_number 设置为零。

```
7.int LIBUSB_CALL libusb_claim_interface(
libusb_device_handle *dev_handle,
int interface_number);
```

请求接口，dev_handle 设备句柄，interface_number 设置为零。

```
8.int LIBUSB_CALL libusb_release_interface(
libusb_device_handle *dev_handle,
int interface_number);
```

释放设备句柄资源，dev_handle 设备句柄，interface_number 设置为零。

```
9.void LIBUSB_CALL libusb_close(
libusb_device_handle *dev_handle);
```

关闭设备句柄，dev_handle 设备句柄。

```
10.void LIBUSB_CALL libusb_exit(
libusb_context *ctx);
```

退出 libusb-1.0 库，ctx 设置为零。

2.2 相机返回数据各位说明

0XD2 会一次性返回相机当前所有的相关信息，0xD2 包含 64 个字节相机状态信息（Camera Statu Information, CSI）

CSI0 ：当前的速度设置，对于 CMOS 相机通常返回值为 CMOS 主频，对于 CCD 相机，返回为 CCD 芯片主频，0 为低于 1M， 1 位 1M 2 为 2M
CSI1..4 ：距离曝光结束的时间（单位微秒） CSI1=MSB CSI4=LSB 部分相机支持该功能
CSI5..8 ：设置的曝光时间
CSI9..11 ：获取固件版本（CSI9：年 CSI10:月 CSI11:日）
CSI12 ：温度类型标识。0=支持摄氏度读出 1=支持 ADU 单位读出 2=支持两者。 如果支持 ADU 单位读出的，在 CSI13..14 CSI15..16 输出，如果支持摄氏度的，在 CSI22..23 CSI24..25：读出
CSI13..14：当前温度（以 ADU 为单位）
CSI15..16：目标温度 （以 ADU 为单位）
CSI17 ：当前 PWM 值
CSI18 ：当前温控模式 1=自动 0=手动
CSI19..21：DDR 当前的存储数据量 CSI19=MSB CSI21=LSB 支持 DDR 的相机支持该功能
CSI22..23：当前温度（以 0.1 摄氏度为单位）
CSI24..25：目标温度（以 0.1 摄氏度为单位）

CSI28..29：输出图像 X 尺寸
CSI30..31：输出图像 Y 尺寸
CSI32 ：输出图像位数
CSI33 ：相机的 USB 端口速度 1=USB1.0 2=USB2.0 3=USB3.0

CSI38..45：8 BYTES 滤镜轮缓冲区 用于接受相机的串口缓冲区内前 8 个字节。
CSI46 ： 相机子型号
CSI47 ：彩色/黑白 0: mono 1: RGB 2: CMYG 3. RGBW
CSI48..63：相机序列号（共 16 字节）

2.3 示例代码

1.获取相机信息

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

#define uchar unsigned char
#define uint  unsigned int

#define TIMEOUT  0
#define USB_VID  0x1618
#define USB_PID  0xc166
#define CTRL_IN  (LIBUSB_REQUEST_TYPE_VENDOR | LIBUSB_ENDPOINT_IN)
#define CTRL_OUT (LIBUSB_REQUEST_TYPE_VENDOR | LIBUSB_ENDPOINT_OUT)
#define USB_RQ   0x04

int main(){
    int i,j,r = 0;
    int actual_length;

    struct libusb_device_handle *dev = NULL;
    struct libusb_device **d = NULL;
    uchar data_recv[0x40] = { 0 };
    uchar data_cmd[2][0x10] =
    {{0xa0,0x01,0x00,0x01,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
     {0xa6,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

    r = libusb_init(NULL);//initiate libusb
    if(r < 0){
        printf("Initial USB lib failed!\n");
        return -1;
    }

    dev = libusb_open_device_with_vid_pid(NULL,USB_VID,USB_PID);//open device
    if(dev == NULL){
        printf("Open device failed!\n");
        return -1;
    }else{
        printf("Open successed!\n");
    }
    r = libusb_claim_interface(dev,0);
    if(r < 0){
        printf("Cannot claim interface!\n");
        return -1;
    }else{
        printf("Claimed interface successed!\n");
    }

    r = libusb_kernel_driver_active(dev,0);
    if(r == 1){
        printf("kernel driver active!\n");
        r = libusb_detach_kernel_driver(dev,0);
        if(r == 0)//detach kernel
            printf("Kernel driver detached!\n");
    }

    for(i = 0;i < 2;i ++){
        printf("i = %d\n",i);
        r = libusb_control_transfer(dev,0x40,0xD1,0,0,data_cmd[i],16,0);
        if(r < 0){
            fprintf(stderr,"Error occered!(%d)\n",r);
            return -1;
        }
        sleep(3);
        r = libusb_control_transfer(dev,0xc0,0xD2,0,0,data_recv,64,0);
        if(r < 0){
            fprintf(stderr,"Error occered!(%d)\n",r);
            return -1;
        }else{
            for (j = 0; j < sizeof(data_recv); j++){
                printf("\033[1;31;40m%2d ",j);
                printf("\033[0m%02x ", data_recv[j]);
                if((j + 1) % 16 == 0)
                    printf("\n");
            }
        }
        printf("\n");
    }

    libusb_release_interface(dev,0);
    libusb_close(dev);
    libusb_exit(NULL);//exit libusb
    free(data);
}
```

2.获取设备的端点号（endpoint）

```
#include <stdio.h>
#include "libusb.h"

static void print_devs(libusb_device **devs){
    libusb_device *dev;
    int i = 0, j = 0;
    struct libusb_config_descriptor *config;
    while((dev = devs[i++]) != NULL){
        struct libusb_device_descriptor desc;

        int r = libusb_get_device_descriptor(dev,&desc);//获取设备描述符
        if(r < 0){
            fprintf(stderr,"Error occered!(%d)\n",r);
            return -1;
        }
        printf("%04x:%04x ",desc.idVendor,desc.idProduct);
        r = libusb_get_active_config_descriptor(dev,&config);//获取端点号描述符
        if(r < 0){
            fprintf(stderr,"Error occered!(%d)\n",r);
            return -1;
        }
        printf("endpoint:0x%x",config->interface->altsetting->endpoint->bEndpointAddress)
;
        printf("\n");
    }
    libusb_free_config_descriptor(config);

}
int main(){
    libusb_device **devs;
    int r;
    ssize_t cnt;
    r = libusb_init(NULL);//初始化 libusb-1.0 库
    if(r < 0)
        return r;
    cnt = libusb_get_device_list(NULL,&devs);//获取 usb 设备列表
    if(cnt < 0)
        return (int)cnt;
    print_devs(devs);//打印输出端点号
    libusb_free_device_list(devs,1);//释放设备列表资源
    libusb_exit(NULL);//退出 libusb-1.0 库
    return 0;
}
```

3.libusb 通信程序

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include "libusb.h"
#include "/usr/local/include/opencv2/highgui/highgui.hpp"
#include "cv.h"
#include "cxcore.h"

#define uchar unsigned char
#define uint unsigned int

#define TIMEOUT 0
#define USB_VID 0x1618
#define USB_PID 0xc166
#define CTRL_IN (LIBUSB_REQUEST_TYPE_VENDOR | LIBUSB_ENDPOINT_IN)
#define CTRL_OUT (LIBUSB_REQUEST_TYPE_VENDOR | LIBUSB_ENDPOINT_OUT)
#define USB_RQ 0x04

void delayms(int xms){
    int i,j;
    for(i = 0;i < xms;i ++){
        for(j = 0;j < 110;j ++){};
    }
}

int main(){
    IplImage* image = cvCreateImage(cvSize(4968,3378),IPL_DEPTH_8U,3);
    IplImage* iplgray = cvCreateImage(cvGetSize(image),IPL_DEPTH_8U,3);
    // IplImage* iplCanny = cvCreateImage(cvSize(4968,3378),IPL_DEPTH_8U,3);
    // IplImage* ipltemp = cvCreateImage(cvGetSize(image_h),IPL_DEPTH_16U,3);
    // IplImage* image = cvCreateImage(cvSize(4968,3378),IPL_DEPTH_16U,1);

    int i,j,r = 0;
    struct libusb_device_handle *dev = NULL;
    struct libusb_device **d = NULL;
    int actual_length;
    struct libusb_transfer *xfr;
    char *data = (char *)malloc(16780000);
    char *re;
    struct libusb_config_descriptor **config;

    //data of receiving from camera
```

```
uchar data_recv[0x40] = { 0 };
uchar data_cmd[2][0x10] =
{{0xa0,0x01,0x00,0x01,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},
{0xa6,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

r = libusb_init(NULL);//initiate libusb
if(r < 0){
    printf("Initial USB lib failed!\n");
    return -1;
}

dev = libusb_open_device_with_vid_pid(NULL,USB_VID,USB_PID);//open device
if(dev == NULL){
    printf("Open device failed!\n");
    return -1;
}else{
    printf("Open successed!\n");
}

r = libusb_claim_interface(dev,0);
if(r < 0){
    printf("Cannot claim interface!\n");
    return -1;
}else{
    printf("Claimed interface successed!\n");
}

r = libusb_kernel_driver_active(dev,0);
if(r == 1){
    printf("kernel driver active!\n");
r = libusb_detach_kernel_driver(dev,0);
if(r == 0)//detach kernel
    printf("Kernel driver detached!\n");
}

for(i = 0;i < 2;i ++){
    printf("i = %d\n",i);
    //send command to comera
    r = libusb_control_transfer(dev,0x40,0xD1,0,0,data_cmd[i],16,0);
    if(r < 0){
        fprintf(stderr,"Error occered!(%d)\n",r);
        return -1;
    }
    sleep(3);
    //receive data from camera
    r = libusb_control_transfer(dev,0xc0,0xD2,0,0,data_recv,64,0);
    if(r < 0){
        fprintf(stderr,"Error occered!(%d)\n",r);
        return -1;
    }else{
        for (j = 0; j < sizeof(data_recv); j++){
            printf("\033[1;31;40m%2d ",j);
            printf("\033[0m%02x ", data_recv[j]);
            if((j + 1) % 16 == 0)
            printf("\n");
        }
    }
    printf("\n");
}

sleep(3);

r = libusb_bulk_transfer(dev,0x81,(uchar *)data,16384,&actual_length,0);
printf("%x %x %x %x\n",data[0],data[1],data[2],data[3]);
bzero(data,sizeof(data));

sleep(3);

for(i = 0;i < 2048;i++){
    delayms(3);
    printf("\033[1;31;40m j l(d) a_l | l(re) || r i || iD[j+0] iD[j+1] iD[j+2]
    | d[0] d[1]       d[2]\n");
    r = libusb_bulk_transfer(dev,0x81,(uchar *)data,16384,&actual_length,0);
    delayms(3);
    re = strcat(image->imageData,data);

    j = strlen(re) - strlen(data);
    printf("\033[0m%8d ",j);

    printf("\033[0m%5d %5d | ",(int)strlen(data),actual_length);
    printf("\033[0m%8d || %d %4d || ",(int)strlen(re),r,i);



    printf("\033[0m%8x %8x %8x | %8x %8x %8x\n",image->imageData[j +
0],image->imageData[j + 1],image->imageData[j + 2],data[0],data[1],data[2]);
    bzero(data,sizeof(data));
    printf("\n");
}
```

```
    printf("%x %x %x | ",image->imageData[0],image->imageData[1],image->imageData[2]);
    // cvCvtColor(image,iplgray,CV_BGR2GRAY);
    // cvSmooth(iplgray,iplCanny,3,3,0,0);
    cvNot(image,iplgray);
    // cvCanny(image,iplCanny,50,150,3);
    printf("%x %x %x\n",iplgray->imageData[0],iplgray->imageData[1],iplgray->imageData[2]
);
    //printf("size = %d\n",(int)sizeof(data));


    sleep(3);
    cvNamedWindow("Source", 1);
    cvShowImage("Source", iplgray);
    cvWaitKey(0);
    cvDestroyWindow("Source");
    cvReleaseImage(&image);
    libusb_release_interface(dev,0);
    libusb_close(dev);
    //libusb_free_transfer(xfr);
    libusb_exit(NULL);//exit libusb
    free(data);
}
```

注：代码只是框架，不完整，需要根据需求自行完善。