QHYCCD SDK instruction manual

Ⅰ.Function Resolution

All of the functional functions that are used in the secondary development are declared in the qhycc.h header file, using only the header file.The following is an introduction to application functions.

1.1 Basic operation function

1.InitQHYCCDResource

1.uint32_t InitQHYCCDResource(void);

Function Description:

Initializing the SDK resources and if the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
int ret = QHYCCD_ERROR;
ret = InitQHYCCDResource();
if(ret == QHYCCD_SUCCESS){
    printf(“Success\n”);
}else{
    printf(“Failed\n”);
}
```

2.uint32_t ReleaseQHYCCDResource();

Function Description:

Release camera resource,and if the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
int ret = QHYCCD_ERROR;
ret = ReleaseQHYCCDResource();
if(ret == QHYCCD_SUCCESS)
    printf(“Release SDK resource success.\n”);
else
    printf(“Release SDK resource failed.\n”);
```

3.uint32_t OSXInitQHYCCDFirmware(char *path);

Parameter Introduce:

path	The path of directory where firmware in,the firmware must be stored in the "firmware" folder.The path recorded in "path" parameter is actually the storage path of "firmware" folder.

Function Description:

This function used to download firmware to camera,only the Mac platform needs this function for the time being,we use driver on Windows platform,and use the 85-qhyccd.rules file and fxload on Linux platform.

Sample Code:

```
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);

Function Description:

This function can scan connected QHYCCD device and return the number of scanned device after executing successfully.

Sample Code:

```
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);

Parameter Introduce:

id	The camera ID returned by GetQHYCCDId();

Function Description:

The camera will be turned on according to the ID returned by GetQHYCCDId, and if the function execute successfully,the function will return camera handle.If the handle is not empty,that mean the function executes successfully.Then all operate functions need this handle as parameter.

Sample Code:

```
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);

Parameter Introduce:

camhandle	The camera handle returned by OpenQHYCCD();

Function Description:

This function can close camera,and disconnect with camera.If the function executes successfully,it will return QHYCCD_SUCCESS。

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();

Function Description:

Initializing camera resource.If this function executes successfully,it will return QHYCCD_SUCCESS。

Sample Code:

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

1.2 Information acquisition function

These functions used to get camera's one or several parameters.You can get some information according these parameters,can also use these parameters be other function's parameter,or according to these parameters to do a judgement.

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

Parameter Introduce:

index	This is the index of the array of camera structure,and must be lower than the equal of ScanQHYCCD(); return value;
id	A pointer variate which is char type,it is used to receive camera ID that returned by function.

Function Description:

Used to get connected camera's ID.If the function executes successfully,it will return QHYCCD_SUCCESS.The ID of each camera is made up of the camera model and serial number.For example, QHY183C-c915484fa76ea7552, the front QHY183C is the camera model, and the rear c915484fa76ea7552 is the serial number of the camera.Each camera has its own serial number, and the serial numbers are different even for the same type of camera.Its role is to distinguish the camera from the camera, which is necessary as a multi-camera test.

Sample Code:

```
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);

Parameter Introduce:

id	The camera's ID returned by GetQHYCCDId();
model	Used to receive and store camera model;

Function Description:

Used to get camera model,for example QHY183C-c915484fa76ea7552,its camera model is QHY183C.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle retuened by OpenQHYCCD();
buf	Used to store the information about firmware version;

Function Description:

Get firmware version,you only need this function on Linux&Mac platform.You can judge if the firmware that used now is the newest by the SDK version.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

year	Used to receive the year of SDK version;
month	Used to receive the month of SDK version;
day	Used to receive the day of SDK version;
subday	Set it to be 0;

Function Description:

Used to get SDK version,you can use this function on all platform,and can judge if the SDK is the newest version.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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	The camera handle returned by OpenQHYCCD();

Function Introduce:

Used to get device type,for example DEVICETYPE_QHY183C(4045).If the function executes successfully,it will return the macro that define in qhyccdcamdef.h.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
controlId	The macro which is represent function parameter;
min	The parameter's minimum;
max	The parameter's maximum;
step	The parameter's step;

Function Description:

Used to get camera parameter's maximum,minimum and step,you can know about camera parameter's range and step(for example gain offset). If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
hd	Used to receive humidity;

Function Description:

Used to get camera humidity,only A several camera and IC16803 has the function temporarily.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
buf	Used to receive camera status;

Function Description:

Used to get camera's status, it include idle, waiting, exposing and read out, only A several camera has this function. If the function executes successfully, it will return QHYCCD_SUCCESS.

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

00	fe	81	74:idle, camera don't expose and readout
01	fe	81	74:waiting, a span before starting expose, very short
02	fe	81	74:exposing, open shutter to start expose, and will close shutter after expose
03	fe	81	74:read out image data

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
chipw	The width of chip;
chiph	The height of chip;
w	The width of image;
h	The height of image;
pixelw	The width of pixel;
pixelh	The height of pixel;
bpp	The bit depth of image;

Function Description:

Used to get camera's chip information. If the function executes successfully, it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
startX	The start position of image effective area in horizontal direction;
startY	The start position of image effective area in vertical direction;
sizeX	The width of image effective area;
sizeY	The height of image effective area;

Function Description:

Used to get effective size and start position.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
startX	The start position of image overscan area in horizontal direction;
startY	The start position of image overscan area in vertical direction;
sizeX	The width of image overscan area;
sizeY	The height of image overscan area;

Parameter Description:

Some CCD have overscan area,this function is used to get the size and start position of image.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
controlId	The macro which is represent function parameter;

Here are some common ID:

```
CAM_COLOR,              //Used to check if camera is color camera
CAM_BIN1X1MODE,         //Used to check if camera has 1X1 bin mode
CAM_BIN2X2MODE,         //Used to check if camera has 2X2 bin mode
CAM_BIN3X3MODE,         //Used to check if camera has 3X3 bin mode
CAM_BIN4X4MODE,         //Used to check if camera has 4X4 bin mode
CAM_MECHANICALSHUTTER,  //Used to check if camera has mechanical shutter
CAM_GPS,                //Used to check if camera has GPS
CONTROL_COOLER          //Used to check if camera is cooler camera
CONTROL_CHANNELS,       //Used to get the channel of image
CONTROL_CURTEMP,        //Used to get the temperature of camera currently
CONTROL_CURPWM,         //Used to get cooler PWM currently
CONTROL_MANULPWM,       //Used to set up color PWM manually
CONTROL_WBR,            //Used to adjust the white balance of red light
CONTROL_WBB,            //Used to adjust the white balance of blue light
CONTROL_WBG,            //Used to adjust the white balance of green light
CONTROL_GAIN,           //Used to adjust camera gain
CONTROL_OFFSET,         //Used to set up camera offset
CONTROL_EXPOSURE,       //Used to set up camera expose time
CONTROL_SPEED,          //Used to set up camera USB transfer speed
CONTROL_TRANSFERBIT,    //Used to get camera bit depth
CONTROL_USBTRAFFIC,     //Used to adjust FPS
```

Parameter Description:

Used to check if camera has some function by macro,if the camera has some function,it will return QHYCCD_SUCCESS,if not,it will return QHYCCD_ERROR.When you use the function to check if camera is color camera(CAM_COLOR),the function will return camera's bayer order if it executes successfully,if not,it will return QHYCCD_ERROR.Camera's bayer orders are defined in qhyccdstruct.h file,its definition as follows:

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

Sample Code:

```
1.Check if camera is cooler camera
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.Check if camera is color camera
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");
```

Note:All macro introduce

The effect of some of the command word is just check whether the camera has a certain function, some command word can be used to check whether the camera have a function, can also set the corresponding parameters of camera, some command word is used for the rest of the SDK within as a symbol of some function parameters or state and use.

```
enum CONTROL_ID
{
    CONTROL_BRIGHTNESS = 0,              //Used to set up image brightness
    CONTROL_CONTRAST,                    //Used to set up image contrast
    CONTROL_WBR,                         //Used to set up the white balance of red light
    CONTROL_WBB,                         //Used to set up the white balance of blue light
    CONTROL_WBG,                         //Used to set up the white balance of green light
    CONTROL_GAMMA,                       //Used to Gamma correction
    CONTROL_GAIN,                        //Used to set up gain
    CONTROL_OFFSET,                      //Used to set up offset
    CONTROL_EXPOSURE,                    //Used to set up expose time(us)
    CONTROL_SPEED,                       //Used to set camera USB transfer speed
    CONTROL_TRANSFERBIT,                 //Used to set up or get image bit depth
    CONTROL_CHANNELS,                    //Used to set up or get image channels
    CONTROL_USBTRAFFIC,                  //Used to set up or get usb traffic
    CONTROL_ROWNOISERE,                  //Used to row noisere
    CONTROL_CURTEMP,                     //Used to get current temperature
    CONTROL_CURPWM,                      //Used to get current PWM
    CONTROL_MANULPWM,                    //Used to set up cooler manually
    CONTROL_CFWPORT,                     //Used to check if camera can connect filter
    CONTROL_COOLER,                      //Used to check if camera is cooler camera
    CONTROL_ST4PORT,                     //Used to check if camera has ST4PORT
    CAM_COLOR,                           //Used to check if camera is color camera
    CAM_BIN1X1MODE,                      //Used to check if camera has 1X1 bin mode
    CAM_BIN2X2MODE,                      //Used to check if camera has 2X2 bin mode
    CAM_BIN3X3MODE,                      //Used to check if camera has 3X3 bin mode
```

```
    CAM_BIN4X4MODE,                     //Used to check if camera has 4X4 bin mode
    CAM_MECHANICALSHUTTER,              //Used to check if camera has mechanical shutter
    CAM_TRIGER_INTERFACE,               //Used to check if camera has triger mode
    CAM_TECOVERPROTECT_INTERFACE,       //temperature protect,limit cooler PWM
    CAM_SINGNALCLAMP_INTERFACE,         //Used to check if camera has signal clamp
    CAM_FINETONE_INTERFACE,             //Used to check if camera has fine tone
    CAM_SHUTTERMOTORHEATING_INTERFACE,  //Used motor to heating shutter
    CAM_CALIBRATEFPN_INTERFACE,         //Used to FPN correction
    CAM_CHIPTEMPERATURESENSOR_INTERFACE, //Used to check if camera has chip temperature
                                        //sensor
    CAM_USBREADOUTSLOWEST_INTERFACE,    //USB read out data as lowest speed

    CAM_8BITS,                          //Used to check if can set camera be 8 bits
    CAM_16BITS,                         //Used to check if can set camera be 16 bits
    CAM_GPS,                            //Used to check if camera has GPS

    CAM_IGNOREOVERSCAN_INTERFACE,       //Ignore overscan area
    QHYCCD_3A_AUTOBALANCE,              //Used to automatically balance
    QHYCCD_3A_AUTOEXPOSURE,             //Used to automatically expose
    QHYCCD_3A_AUTOFOCUS,                //Used to automatically focus
    CONTROL_AMPV,                       //Used to check if camera has AMPV
    CONTROL_VCAM,                       //The switch of virtual camera
    CAM_VIEW_MODE,                      //Used to vieew mode

    CONTROL_CFWSLOTSNUM,                //Used to check filter slot number
    IS_EXPOSING_DONE,                   //Used to check if camera has exposed
    ScreenStretchB,                     //Used to screen stretch
    ScreenStretchW,                     //Used to screen stretch
    CONTROL_DDR,                        //Used to check if camera has DDR
    CAM_LIGHT_PERFORMANCE_MODE,         //HGC/LGC gain control(has been combined in gain
                                        //control)

    CAM_QHY5II_GUIDE_MODE,              //Used to guid
    DDR_BUFFER_CAPACITY,                //Used to check DDR capacity
    DDR_BUFFER_READ_THRESHOLD           //The threshold value of DDR read,start to read
                                        //out image data if more then this
};
```

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

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
controlId	The macro which is represent function

Function Description:

This function can get camera information according to CONTROL_ID,for example, expose time,gain,offset.If the function executes successfully,it will returnQHYCCD_ERROR.

Sample Code:

```
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 Function Control Function

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

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
controlId	The macro which is represent function;
value	Used to receive the value of parameter;

Function Description:

Used to set up camera parameter according to controlId.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
1.Set up expose time
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.Set camera traffic
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.Set camera gain
ret = SetQHYCCDParam(camhandle,CONTROL_GAIN,15);
if(ret == QHYCCD_SUCCESS)
    printf("Set camera gain success.\n");
else
    printf("Set camera gain failed.\n");
4.Set camera offset
ret = SetQHYCCDParam(camhandle,CONTROL_OFFSET,140);
if(ret == QHYCCD_SUCCESS)
    printf("Set camera gain success.\n");
else
    printf("Set camera gain failed.\n");
5.Set camera transfer speed
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.temperature control
#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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
targettemp	CCD target temperature;

Function Description:

Used to control camera cooler,it is same as SetQHYCCDParam(CONRTOL_COOLER).If the function executes successfully,it will return QHYCCD_SUCCESS.If you don't if your camera is cooler camera,you can use IsQHYCCDControlAvailable(CONTROL_COOLER) to check it.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
onoff	The switch of color mode,1:open,0:close;

Function Description:

Used to set up color camera mode open or close,it is only effective for color camera,you need make sure your camera is color camera.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
wbin	The bin in horizontal direction;

hbin	The bin in vertical direction;

Function Description:

Used to set camera bin mode,for example 1X1,2X2 etc.You can use IsQHYCCDControlAvailable(); to check which bin mode is supported by your camera.The function need use with SetQHYCCDResolution();If the camera executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

See SetQHYCCDResolution(); sample code to get more detail.

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

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
x	Set it to be 0;
y	Set it to be 0;
xsize	The width of image;
ysize	The height of image;

Function Description:

Used to set up camera's resolution,need use with SetQHYCCDBinMode();If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
mode	The stream mode of camera,0:single frame,1:live frame;

Function Description:

Set up camera stream mode,you can chose single frame mode or live frame mode.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();

Function Description:

Used to get the length of image data,you can create a memory space use this length

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();

Function Description:

Used to start to expose a frame,you can use SetQHYCCDParam(CONTROL_EXPOSURE) to set up expose time,its unit is us.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
w	The width of image;
h	The height of image;
bpp	The bit depth of image;
channels	The channel of image;
imgdata	Used to receive image data;

Function Description:

Used to get a frame image data,and store data in "ImgData" variate.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();

Function Description:

Stopping camera exposing and stop camera read out.If the function executes successfully,it will return QHYCCD_SUCCESS.When stopping,you need make sure that the software is synchronized with the camera, the camera does not output data and the software does not receive data,or the camera outputs the data and the software receives the data, otherwise the software or the cameras will die.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();

Function Description:

Stopping camera exposing,but doesn't stop read out.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();

Function Description:

Starting live frame mode exposing,and the camera will generate image data sostenuto,meanwhile,PC should also read out image data and display.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
w	The width of image;
h	The height of image;
bpp	The bit depth of image;
channels	The channel of image;
ImgData	Used to receive image data;

Function Description:

Used to get image data in live frame mode,and store in "ImgData".If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();

Function Description:

Stopping camera live frame mode.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
x	The virtual width of image,it's imagew/bin;
y	The virtual height of image,it's imageh/bin;
InBuf	Image data that get by GetQHYCCDSingleFrame();

Function Description:

Get histogram information according to image data.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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 Code：

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();

Function Description:

Used to get the reading progress of image data,only QHY23 and A series camera has this function temporarily,if the function executes successfully,it will return reading progress,otherwise will return QHYCCD_ERROR.You need use this function before ExpQHYCCDSingleFrame();,and because the return value is estimated,it may be not very correct,even more 100%.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
value	Used to control triger function enable or not,1:enable,0:disable;

Function Description:

Used to set up camera trigger function.If triggered, the camera does not immediately begin to expose but wait until the outside trigger signal arrives.In the case of camera trigger, if enable trigger, it can exit the waiting state.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();

Function Description:

Used to check if filter has been connected,onle QHY5IIICOOL series camera and MINICAM5F_M has this function,must make sure filter has been powered on,otherwise it will return QHYCCD_ERROR.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
status	Used to receive current filter position;

Function Description:

Used to get filter status(which position),its range is 0 to the number of filter wheel holes subtract 1,for example,if the filter wheel hole number is 8,its range is 0~7.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
order	The target position of filter wheel;
length	The length of "order";

Function Description:

Used to control filter wheel.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

loglevel	The level of log output;

Function Description:

Used to control program output log information on terminal,it will output different log information,0:LOG_DEBUG,1:LOG_TRACE.

Sample Code:

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

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

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
i	Used to control VCOX frequency,its range is 0~4095;

Function Description:

Used to control GPS camera VCOX frequency.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
i	Used to set up LED enable or not,0:disable,1:enable;

Function Description:

Used to control LED calibration.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
i	Used to set up camera master/slave mode,0:master,1:slave;

Function Description:

Used to control GPS camera master/slave mode. If the function executes successfully, it will return QHYCCD_SUCCESS. When camera is in slave mode, you can use SetQHYCCDGPSSlaveModeParameter(); to set up parameter, the target_sec"JS" that defined by QHYCCD, it mean a time. If the function executes successfully, it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
is_slave	Depending on GPS camera mode,0:master,1slave;
pos	Used to set up LED pulse position;
width	Used to set up LED pulse width;

Function Description:

Used to set up LED pulse for shutter expose. If you changed expose time, you must set up this position. The measurement circuit will use this position as the shutter starting time.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
is_slave	Depending on GPS camera mode,0:master,1slave;
pos	Used to set up LED pulse position;
width	Used to set up LED pulse width;

Function Description:

When you changed the exposure. You must set this position. The measurement circuit will use this position as the shutter ending time.

Sample Code:

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

ADD:Data Structure of the Image Head

The camera records the GPS information and insert into each frame's head. This function can be enabled/disable by the API:

```
enable :ret=SetQHYCCDParam(g_hCam,CAM_GPS,1);
disable: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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
InputData16	Input 16 bits image data;
OutputData8	Output 8 bits image data;
imageX	The width of image;
imageY	The height of image;
B	Used to set up gray stretch;
W	Used to set up gray stretch;

Function Description:

Change 16 bits image data to 8 bits image data,meanwhile do gray stretch.

Sample Code:

```
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);

Parameter Introduce:

handle	The camera handle returned by OpenQHYCCD();
focusCenterX	The X position of focus center;
focusCenterY	The Y position of focus center;

Function Description:

Used to set up focus,it has different ways of setting up according to different BIN&ROI.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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);

Function Description:

Corresponding QHY9 and QHY11,used to optimize the drive time sequence of CCD, which can further optimize the readout noise of CCD.As this function is more complex, please contact our software engineer if necessary.

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

Parameter Introduce:

vid	The VID of camera;
pid	The PID of camera;
imgpath	The position of firmware;

Function Description:

Downloading firmware to camera.If the function executes successfully,it will return QHYCCD_SUCCESS.

Sample Code:

```
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 Example Program

1.Single Frame Sample

```
#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.Live Frame Sample

```
#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);
    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.Camera temperature control

```
#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);
sleep(80);

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);//170000000);
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);
if(ret == QHYCCD_SUCCESS)
{
    printf("SetQHYCCDResolution success!\n");
}
else
{
    printf("SetQHYCCDResolution failed!\n");
    goto failure;
}

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

```
    goto failure;
}

uint32_t length = GetQHYCCDMemLength(camhandle);
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);
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);
    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);
    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;
    }
    printf("QHYCCD | SingleFrameSample.cpp | end\n");
    return 0;

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

II.Low Level Protocol

Camera low level protocol:http://qhyccd.com/bbs/index.php?board=24.0
libusb official website:http://libusb.info/

2.1 Function Description

Linux&Mac:

Linux&Mac use libusb and low level protocol to communicate with camera,here are some common functions:

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

Initializing function,used to initialize libusb-1.0 library,must use it in the front of program,its parameter is NULL commonly.

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

Used to open USB device,if the function executes successfully,can get the handle of USB device,Set up ctx to NULL, vendor_id and product_id is the VID and PID of USB device.

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);

Control transfer function,used to send order or read out data from camera,"dev_handle" is device handle,request_type&bRequest used to set up transfer direction.
send order to camera :0x40,0xD1
read data from camera:0xc0,0xD2

"data" is the order you want to send or the data you want to read,"wLength" is the length of "data","timeout" is overtime threshold value.

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);

Bulk transfer function,used to get image data from camera,"dev_handle" is device handle,"endpoint" is camera end point,you can get it by program,"data" is used to get image

data,"length" is the length of "data","actual_length" is the actual length that you get from camera,"timeout" is overtime threshold value.

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

Used to judge if has device driver,"dev_handle" is the handle of device,"interface_number" set it to be 0.

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

Remove the device driver,"dev_handle" is the handle of device,"interface_number" set it to be 0.

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

Claim interface,"dev_handle" is the handle of device,"interface_number" set it to be 0.

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

Release device hundle resource,"dev_handle" is the handle of device,"interface_number" set it to be 0.

9.void LIBUSB_CALL libusb_close(
libusb_device_handle *dev_handle);

Close device handle,"dev_handle" is the handle of device.

10.void LIBUSB_CALL libusb_exit(
libusb_context *ctx);

Exit libusb-1.0 library,"ctx" set it to be 0.

2.2 instructions about the data returned by camera

Using 0xD2 parameter will return all relevant information of the camera at once.0xD2 contains 64 byte Camera Status Information(CSI).

CSI0 :For the current speed setting, CMOS camera usually returns a CMOS main frequency, and for CCD camera, the main frequency of the CCD chip is returned, 0 is less than 1M, and 1 is 1M, 2 is 2M;
CSI1..4 :The remaining time of expose,its unit is us.Only part camera supports this function;
CSI5..8 :Expose time;
CSI9..11 :SDK version(CSI9:year CSI10:month CSI11:day);
CSI12 :Temperature type identification.0 = support Celsius read,1 = support ADU unit read, 2 = support both.If is the ADU unit read out, in CSI13..14&CSI15..16 output,if is the Celsius read out,in CSI22..23&CSI24..25 out put;
CSI13..14:Current temperature(ADU);
CSI15..16:Target temperature(ADU);
CSI17 :Current PWM;
CSI18 :Current temperature control mode,1:auto,0:manually;
CSI19..21:The length of data in DDR,CSI19 = MSB,CSI21 = LSB;
CSI22..23:Current temperature,its unit is 0.1℃;
CSI24..25:Target temperature,its unit is 0.1℃;

CSI28..29:The width of image;
CSI30..31:The height of image;
CSI32 :The bit depth of image;
CSI33 :The type of usb,1:USB1.0,2:USB2.0,3:USB3.0;

CSI38..45:The 8 BYTES filter wheel buffer is used to accept the first 8 BYTES in the camera's serial port buffer.
CSI46 :The sub model of camera;
CSI47 :Color/Mono,0:mono,1:RGB,2:CMYG,3:RGBW;
CSI48..63:The serial number(16byte);

2.3 Example Program

1.Get information from camera

```
#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);

        //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");
    }

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

2.Get device 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->bEndpointAddres
s);
        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.Control camera take a frame image

```
#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);
}
```

Note:These demo just is a sample,you can enrich them by your needs.