PlayerOne Software Development Kit Manual

Version:1.1

2020.12.11

1. Introduction

```
This software development kit (SDK) defines a set of data types and functions that
are used to operate PlayerOne's camera, the API interfaces are C language
interface, so C and C++ projects can directly use these API interfaces. You can
also use other languages (such as C#, python) to re-encapsulate our API. For now,
this SDK can only run under windows, and the version under linux and Mac OS will
be coming soon.

Header file: PlayerOneCamera.h, note: there are detailed explanations of types and
functions in this file.

Windows: import library and dynamic library: PlayerOneCamera.lib,
PlayerOneCamera.dll.note: please put the PlayerOneCamera.dll in the folder
containing the application’s executable,or add the DLL's path to the system
environment.

Linux: coming soon.

Mac OS: coming soon.
```

2. Get the count and properties of cameras

```
int camera_count = POAGetCameraCount();
POACameraProperties **ppPOACamProp =
(POACameraProperties **)malloc(sizeof(POACameraProperties *) * camera_count);

int i;
for(i = 0; i < camera_count; i++)
{
    //malloc the POACameraProperties memory
    ppPOACamProp[i] = (POACameraProperties *)malloc(sizeof (POACameraProperties));
    POAErrors error = POAGetCameraProperties(i, ppPOACamProp[i]); //get camaera
properties
    if(error == POA_OK)
    {
        //print camera ID and name
        printf("camera ID: %d, camera name: %s \n", ppPOACamProp[i]->cameraID,
ppPOACamProp[i]->cameraModelName);
    }
    else
    {
        printf("get camera properties failed, index: %d, error code: %s \n", i,
POAGetErrorString(error));
    }
}
```

3. Open camera

```
POAErrors error = POAOpenCamera(ppPOACamProp[0]->cameraID);
if(error != POA_OK)
{
    //Handling errors
}
```

4. Initialize camera

```
POAErrors error = POAInitCamera(ppPOACamProp[0]->cameraID);
if(error != POA_OK)
{
    //Handling errors
}
```

5. Get the configs and their attributes supported by the camera

```
int config_count = 0;
POAErrors error = POAGetConfigsCount(ppPOACamProp[0]->cameraID, &config_count);
if(error != POA_OK)
{
    //Handling errors
    printf("Get config count failed! , error code: %s \n",
POAGetErrorString(error));
}

POAConfigAttributes **ppConfAttr = (POAConfigAttributes
**)malloc(sizeof(POAConfigAttributes *) * config_count);

for(i = 0; i < config_count; i++)
{
    ppConfAttr[i] = (POAConfigAttributes *)malloc(sizeof (POAConfigAttributes));

    POAErrors error = POAGetConfigAttributes(ppPOACamProp[0]->cameraID, i,
ppConfAttr[i]);

    if(error == POA_OK)
    {
        printf("\n");
        printf("config name: %s, config description: %s \n", ppConfAttr[i]-
>szConfName, ppConfAttr[i]->szDescription);

        printf("is writable: %d \n", (int)ppConfAttr[i]->isWritable);

        printf("is readable: %d \n", (int)ppConfAttr[i]->isReadable);

        if(ppConfAttr[i]->valueType == VAL_INT)
        {
            printf("min: %ld, max: %ld, default: %ld \n",
            ppConfAttr[i]->minValue.intValue, ppConfAttr[i]->maxValue.intValue,
ppConfAttr[i]->defaultValue.intValue);
        }
        else if(ppConfAttr[i]->valueType == VAL_FLOAT)
        {
            printf("min: %lf, max: %lf, default: %lf \n",
            ppConfAttr[i]->minValue.floatValue, ppConfAttr[i]-
>maxValue.floatValue, ppConfAttr[i]->defaultValue.floatValue);
        }
        else if(ppConfAttr[i]->valueType == VAL_BOOL)
        {// The maxValue and minValue values of this VAL_BOOL type are meaningless
            printf("default is on: %d \n",  (int)ppConfAttr[i]-
>defaultValue.boolValue);
        }
    }
    else
    {
        printf("get config attributes failed, index: %d, error code: %s \n", i,
POAGetErrorString(error));
    }
}
```

note: If a POAConfig's 'valueType' is VAL_BOOL, the 'maxValue' and 'minValue' are meaningless.

6. Set config value

note:If the POAConfig don't support auto, 'isAuto' parameter will be ignored.

1. set config with VAL_INT type

such as setting exposure:

```
int exposure_us = 100000; //100ms
POAConfigValue exposure_value;
POABool isAuto = POA_FALSE;
exposure_value.intValue = exposure_us;
POAErrors error = POASetConfig(ppPOACamProp[0]->cameraID, POA_EXPOSURE,
exposure_value, isAuto);

if(error != POA_OK)
{
    ////Handling errors
}
```

2. set config with VAL_FLOAT type

for now, there is no such a VAL_FLOAT config to set:

```
double doubleVal = 3.14;
POAConfigValue double_value;
double_value.floatValue = doubleVal;
POAErrors error = POASetConfig(ppPOACamProp[0]->cameraID, POA_XXX, double_value,
POA_FALSE);

if(error != POA_OK)
{
    ////Handling errors
}
```

3. set config with VAL_BOOL type

such as setting hardware bin:

```
POABool boolVal = (POABool)1;
POAConfigValue hardbin_value;
hardbin_value.boolValue = boolVal;
POAErrors error = POASetConfig(ppPOACamProp[0]->cameraID, POA_HARDWARE_BIN,
hardbin_value, POA_FALSE);

if(error != POA_OK)
{
    ////Handling errors
}
```

note:If the POAConfig don't support write, POASetConfig function will return POA_ERROR_CONF_CANNOT_WRITE.

7. Get config value

1. get config with VAL_INT type

such as getting gain:

```
POAConfigValue gain_value;
POABool isAuto;
POAErrors error = POAGetConfig(ppPOACamProp[0]->cameraID, POA_GAIN, &gain_value,
&isAuto);

if(error != POA_OK)
{
    ////Handling errors
}
else
{
    long gian = gain_value.intValue;
    int bIsAuto = (int)isAuto; // bool bIsAuto = isAuto == POA_TRUE ? true :
```

```
false;
}
```

2. get config with VAL_FLOAT type

for example, get camera temperature:

```
double fTemp;
POAConfigValue temp_value;
POABool isAuto;
POAErrors error = POAGetConfig(ppPOACamProp[0]->cameraID, POA_TEMPERATURE,
&temp_value, &isAuto);

if(error != POA_OK)
{
    ////Handling errors
}
else
{
    fTemp = temp_value.floatValue;
}
```

3. get config with VAL_BOOL type

for example, get cooler status:

```
POAConfigValue cooler_value;
POABool isAuto;
POAErrors error = POAGetConfig(ppPOACamProp[0]->cameraID, POA_COOLER,
&cooler_value, &isAuto);

if(error != POA_OK)
{
    ////Handling errors
}
else
{
    int bIsCoolerOn = (int)cooler_value.boolValue; // bool bIsCoolerOn =
cooler_value.boolValue == POA_TRUE ? true : false
}
```

8. Set and get image parameters

1. set image parameters

note: if you want to change resolution, if exposing, please stop it first.

```
//set resolution 800 * 480
POAErrors error = POASetImageSize(ppPOACamProp[0]->cameraID, 800, 480); //make
sure width % 4 == 0, height % 2 == 0
if(error != POA_OK)
{
    ////Handling errors
}

// set start position (100, 50)
POAErrors error = POASetImageStartPos(ppPOACamProp[0]->cameraID, 100, 50);
if(error != POA_OK)
{
    ////Handling errors
}
```

2. get image parameters

```
//get image width and height
int width, height;
POAErrors error = POAGetImageSize(ppPOACamProp[0]->cameraID, &width, &height);
if(error != POA_OK)
{
    ////Handling errors
}

//get start position
int startX, startY;
POAErrors error = POAGetImageStartPos(ppPOACamProp[0]->cameraID, &startX,
&startY);
if(error != POA_OK)
{
    ////Handling errors
}
```

9. Set and get image format

1.set image format

note: if you want to change image format, if exposing, please stop it first.

```
POAImgFormat imgFmt = POA_RAW16; // set image format to Raw16(16bit)
POAErrors error = POASetImageFormat(ppPOACamProp[0]->cameraID, imgFmt); //default
image format is POA_RAW8
if(error != POA_OK)
{
    ////Handling errors
}
```

2.get image format

```
POAImgFormat imgFmt;
POAErrors error = POAGetImageFormat(ppPOACamProp[0]->cameraID, &imgFmt);
if(error != POA_OK)
{
    ////Handling errors
}
```

10. Set and get bin method

*what is bin? Bin usually has four options: Bin1, Bin2, Bin3, and Bin4. The number after Bin indicates that several pixels are combined into one pixel. For example, Bin2 indicates that 4 pixels are combined into one pixel (2*2=4). Bin3 means that 9 pixels are combined into one pixel (3*3=9), and Bin1 means no pixel combination.*

1. set bin

note: if you want to change bin, if exposing, please stop it first.

Set Image Bin, the image size (width & height) and start position will be changed, Please call POAGetImageStartPos and POAGetImageSize to get the new image size and start position after binning.

```
//set bin to 2, note: after setting bin, please get the image size and start
position
POAErrors error = POASetImageBin(ppPOACamProp[0]->cameraID, 2); // set bin to 2,
default bin is 1

if(error != POA_OK)
{
    printf("set bin failed, error code: %s \n", POAGetErrorString(error));
}

int startX = 0, startY = 0;
int width = 0, height = 0;

//after setting bin, get the image start position
error = POAGetImageStartPos(ppPOACamProp[0]->cameraID, &startX, &startY);
if(error != POA_OK)
{
    // if get image start postion failed, set startX and startY to 0
    startX = 0;
    startY = 0;
    printf("Get Image Start Pos failed, error code: %s \n",
POAGetErrorString(error));
}

//after setting bin, get the image size
error = POAGetImageSize(ppPOACamProp[0]->cameraID, &width, &height);
if(error != POA_OK)
{
    // if get image size failed, set width and height to the maximum value under
current bin
    width = ppPOACamProp[0]->maxWidth / 2; // Maximum width under current bin
    height = ppPOACamProp[0]->maxHeight / 2; // Maximum height under current bin
    printf("Get Image Size failed, error code: %s \n", POAGetErrorString(error));
}
```

2. get bin

```
int bin;
POAErrors error = POAGetImageBin(ppPOACamProp[0]->cameraID, &bin);
if(error != POA_OK)
{
    ////Handling errors
}
```

11. Get image data

1. start exposure

① Not single frame, alias Vdieo Mode, continuously exposure, it is suitable for short exposure

```
POAErrors error = POAStartExposure(ppPOACamProp[0]->cameraID, POA_FALSE); //
continuously exposure(Video Mode)

if(error != POA_OK)
{
    ////Handling errors
}
```

② Single frame, alias Snap Mode, single exposure, when the exposure is completed, the exposure will not continue, it is suitable for long exposure

```
POAErrors error = POAStartExposure(ppPOACamProp[0]->cameraID, POA_TRUE); // single
exposure(Snap Mode)

if(error != POA_OK)
```

```
{
    ////Handling errors
}
```

2. get image data

```
unsigned long buffer_size = width * height * pixelBytes; //POA_RAW8: pixelBytes =
1, POA_RAW16: pixelBytes = 2, POA_RGB24: pixelBytes = 3
unsigned char* data_buffer = (unsigned char*) malloc(buffer_size); //Memory must
be allocated first
```

note: this is recommended to do in another thread.

① Not Single Frame (Vdieo Mode)

```
while(1)
{
    POABool pIsReady = POA_FALSE;
    while(pIsReady == POA_FALSE)
    {
        if(bIsStop) //Triggered by external conditions, such as clicking a button
        {break;}
        //sleep(exposure_us /1000 / 10); //ms
        POAImageReady(ppPOACamProp[0]->cameraID, &pIsReady);
    }

    if(bIsStop) ////Triggered by external conditions, such as clicking a button
    {break;}

    // mutex.lock: It is recommended to use a lock to protect data_buffer
    error = POAGetImageData(ppPOACamProp[0]->cameraID, data_buffer, buffer_size,
exposure_us /1000 + 500);
    // mutex.unlock
    if(error != POA_OK)
    {
        ////Handling errors
        printf("get image data failed, error code: %s \n",
POAGetErrorString(error));
        continue;
    }
}
```

② Single Frame(Snap Mode)

```
POACameraState cmeraState;
do
```

```
{
    // sleep(exposure_us /1000 / 10); //ms
    // if(breakTrigger)
    // {
    //     break;
    // }
    POAGetCameraState(ppPOACamProp[0]->cameraID, &cmeraState);
}while(cmeraState == STATE_EXPOSING);

POABool pIsReady = POA_FALSE;

POAImageReady(ppPOACamProp[0]->cameraID, &pIsReady);

if(pIsReady == POA_TRUE)
{
    printf("single frame exposure success \n");
    error = POAGetImageData(ppPOACamProp[0]->cameraID, data_buffer, buffer_size,
exposure_us /1000 + 500);
    if(error != POA_OK)
    {
        printf("get image data failed, error code: %s \n",
POAGetErrorString(error));
    }
}
else
{
    printf("single frame exposure failed \n");
}
```

3. stop exposure

```
POAErrors error = POAStopExposure(ppPOACamProp[0]->cameraID);

if(error != POA_OK)
{
    ////Handling errors
}
```

12. Close Camera

```
POAErrors error = POACloseCamera(ppPOACamProp[0]->cameraID); //close camera and
free memory

if(error != POA_OK)
{
    ////Handling errors
}
```