

# **Berxel SDK Development Documentation**

## Contents

1. Abstract.....	6
1.1. SDK Introduction .....	6
1.2. SDK Compatibility .....	6
2. Description of Development kit .....	6
2.1. SDK Development Kit Module .....	6
2.2. SDK Sample Program .....	7
2.2.1. Sample Program Description.....	7
2.3. Call SDK API Steps.....	7
2.3.1. Call Windows SDK API .....	7
2.3.2. Call Linux SDK API.....	7
3. SDK Interface Description .....	8
3.1. BernelHawkContext Module Description.....	8
3.1.1. getBernelContext.....	8
3.1.2. destroyBernelContext.....	8
3.1.3. getDeviceList .....	8
3.1.4. openDevice .....	9
3.1.5. closeDevice .....	9
3.1.6. setDeviceStateCallback .....	10
3.2. BernelHawkDevice Module Description.....	10
3.2.1. startStreams.....	10
3.2.2. startStreams.....	11

## Documentation

3.2.3. stopStreams .....	11
3.2.4. getSupportFrameModes .....	11
3.2.5. setFrameMode .....	12
3.2.6. getCurrentFrameMode .....	12
3.2.7. readColorFrame .....	13
3.2.8. readDepthFrame .....	13
3.2.9. readIrFrame .....	14
3.2.10. readLightIrFrame .....	14
3.2.11. releaseFrame .....	15
3.2.12. startUpgrade .....	15
3.2.13. convertDepthToPointCloud .....	15
3.2.14. getVersion .....	16
3.2.15. getCurrentDeviceInfo .....	16
3.2.16. getCameraIntriscParams .....	17
3.2.17. getDeviceIntriscParams .....	17
3.2.18. getNetParams .....	17
3.2.19. setStreamMirror .....	18
3.2.20. setRegistrationEnable .....	18
3.2.21. setStreamFlagMode .....	19
3.2.22. setSystemClock .....	19
3.2.23. setNetParams .....	19
3.2.24. reconnectNetDevice .....	20

## Documentation

3.2.25. setDenoiseStatus .....	20
3.2.26. openNoiseFilter .....	20
3.2.27. setTemperaTureCompensationStatus .....	21
3.2.28. setFrameSync.....	21
3.2.29. enableColorAutoExposure .....	21
3.2.30. setColorExposureGain .....	22
3.2.31. getColorExposureGain .....	22
3.2.32. setDepthAESTatus .....	22
3.2.33. getDepthAESTatus .....	23
3.2.34. setDepthGain .....	23
3.2.35. getDepthGain.....	23
3.2.36. setDepthExposure.....	23
3.2.37. getDepthExposure .....	24
3.2.38. setDepthElectricCurrent .....	24
3.2.39. getDepthElectricCurrent .....	24
3.2.40. setDepthCloseRangeDefaultGainAndExposure .....	25
3.2.41. enableDeviceSlaveMode.....	25
3.2.42. getDeviceMasterSlaveMode .....	25
3.2.43. setColorQuality .....	26
3.2.44. getColorQuality.....	26
3.2.45. setDepthConfidence .....	26
3.2.46. getDepthConfidence.....	26

## Documentation

3.2.47. enableHightFpsMode .....	27
3.2.48. setSpatialDenoiseStatus .....	27
3.2.49. setTemporalDenoiseStatus .....	27
3.2.50. enableDecodeColorData .....	28
3.3. BeroxelHawkFrame Module Description .....	28
3.4. BeroxelHawkDefines Module Description .....	29
3.4.1. BeroxelHawkPixelType Enumeration Description .....	29
3.4.2. BeroxelHawkStreamType Enumeration Description .....	29
3.4.3. BeroxelHawkStreamFlagMode Enumeration Description .....	29
3.4.4. BeroxelHawkDeviceInfo Structure Description .....	31
3.4.5. BeroxelHawkStreamFrameMode Structure Description .....	31
3.4.6. BeroxelHawkCameraIntrinsic Structure Description .....	31
3.4.7. BeroxelHawkDeviceIntrinsicParams Structure Description .....	32
4. SDK Guidelines .....	33
4.1. Call SDK API Flowchart .....	33
4.2. Get Color Frame .....	33
4.3. Get Depth Frame .....	34
4.4. Get Color and Depth Mix Frame .....	34
4.5. Get Version .....	34
4.6. Get Current Device Info .....	34
4.7. Depth Raw Data Converted To The Distance Value .....	34
4.7.1. BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D .....	35
4.7.2. BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D .....	35
4.8. Depth Raw Data Converted To Point Cloud .....	35
4.9. Set Device Status Monitoring .....	35
4.10. Get Device Intrinsic Params .....	36

4.11. SDK Error Code .....	37
----------------------------	----

## 1. Abstract

### 1.1. SDK Introduction

The Bernel SDK is a software development kit based on the Bernel 3D camera. The product can be widely used in fields where 3D images are required, such as industrial control, consumer electronics, and other related areas. The SDK supports the Android, Windows, Linux, and ROS platforms.

### 1.2. SDK Compatibility

- Windows: Supports Windows 7 and above systems.
- Linux: supports Ubuntu 14 and above. Support Arm x86/x64.
- Supports USB 2.0, requires 2A current.
- 4G RAM is required at least.
- The processor frequency must be at least 2.2 GHz.

## 2. Description of Development kit

### 2.1. SDK Development Kit Module

Module Name	Description
Document	Bernel SDK Development Documentation
Driver	Device DFU mode driver, mainly used for upgrading

## Documentation

Include	SDK header files
Lib	SDK Library, Calling SDK API needs to include the BernelHawk.lib static library
Samples	Sample program source code
Thirdparty	Third party libraries required by SDK sample programs
Tools	Demo Tools
C#	C# SDK

## 2.2. SDK Sample Program

### 2.2.1. Sample Program Description

Sample name	Description
HawkColor	Demonstrate the process of obtaining color frame
HawkDepth	Demonstrate the process of obtaining depth frame
HawkIr	Demonstrate the process of obtaining infrared frame
HawkMixColorDepth	Demonstrate the process of obtaining color and depth frame (resolution is 640 * 400)
HawkMixHDColorDepth	Demonstrate the process of obtaining color and depth frame (resolution is 1280 * 800)
HawkMixColorDepthMatting	Demonstrate registration function

## 2.3. Call SDK API Steps

### 2.3.1. Call Windows SDK API

- Contains the BernelHawk.lib static library
- Include all header files under the Include folder
- Copy all dynamic library files under the lib folder to the project compilation output directory

### 2.3.2. Call Linux SDK API

- Link to the BernelHawk.so in the libs folder
- export LD\_LIBRARY\_PATH=\$LD\_LIBRARY\_PATH:xxx ,xxx indicates the path of

## 3. SDK Interface Description

### 3.1. BernelHawkContext Module Description

The “BernelHawkContext” module is mainly used to maintain a device list and open or close device. Please refer to the header file “BernelHawkContext.h”.

#### 3.1.1. getBernelContext

**[Description]**

Gets the instance of the BernelHawkContext object.

**[Function]**

```
static BernelHawkContext* getBernelContext()
```

**[Params]**

NONE

**[Return value]**

The instance of the BernelHawkContext object: Successfully created

NULL: Failed to create

#### 3.1.2. destroyBernelContext

**[Description]**

Destroy the instance of the BernelHawkContext object.

**[Function]**

```
static void destroyBernelContext(BernelHawkContext* &pBernelContext)
```

**[Params]**

pBernelContext [in] : A pointer to an instance of the BernelHawkContext object.

**[Return value]**

NONE

#### 3.1.3. getDeviceList

**[Description]**

Get a list of devices.



## Documentation

**[Function]**

```
int32_t getDeviceList(BernelHawkDeviceInfo** pDeviceList, uint32_t*  
pDeviceCount)
```

**[Params]**

pDeviceList [out]: A list of devices.  
pDeviceCount [out]: The count of devices.

**[Return value]**

0 : Success  
Other: Failed

**[Remarks]**

The BernelHawkDeviceInfo structure describes detailed device information.  
Please refer to the header file “BernelHawkDefine.h”.

### 3.1.4. openDevice

**[Description]**

Open device.

**[Function]**

```
BernelHawkDevice* openDevice(const BernelHawkDeviceInfo deviceInfo)
```

**[Params]**

deviceInfo[in] : The structure of the device that needs to be opened, and  
can be obtained by the function “getDeviceList”.

**[Return value]**

The instance of the BernelHawkDevice object: The device opened  
successfully.

NULL: Failed to open the device.

**[Remarks]**

BernelHawkDevice is the class of the device management module, please refer  
to the header file “BernelHawkDevice.h”.

BernelHawkDeviceInfo is a structure about device information, please refer to  
the header file “BernelHawkDefines.h”.

### 3.1.5. closeDevice

**[Description]**

Close device.

**[Function]**

```
int32_t closeDevice((BernelHawkDevice* hawkDevice)
```

**[Params]**

hawkDevice[in] : Handle to the currently open device.

**[Return value]**

0 : Success

### 3.1.6. setDeviceStateCallback

**[Description]**

Registers a callback of the state of a device to listen for the state of the device being disconnected or connected.

**[Function]**

```
int32_t setDeviceStateCallback(BernelHawkDeviceStatusChangeCallback  
callback, void* pUserData)
```

**[Params]**

callback[in] : A callback function for the state of the device.

pUserData[in]: A pointer to the user class.

**[Return value]**

0 : Success

Other: Failed

**[Remarks]**

The function “BernelHawkDeviceStatusChangeCallback” is a callback function for device state, please refer to the header file “BernelHawkDefines.h”.

## 3.2. BernelHawkDevice Module Description

The “BernelHawkDevice” module is mainly used to maintain the related operations of the device, including open/close the stream, read the data from the device and get the device information, please refer to the header file “BernelHawkDevice.h”.

### 3.2.1. startStreams

**[Description]**

Open streams.

**[Function]**

```
int32_t startStreams(int streamFlags)
```

**[Params]**

streamFlags[in] : The value of the type of stream you want to open.

**[Return value]**

0: Success

Other: Failed

**[Remarks]**

“BernelHawkStreamType” is an enumeration of stream types, please refer to the section 3.4.2.

### 3.2.2. startStreams

**[Description]**

Open streams.

**[Function]**

```
int32_t startStreams(int streamFlags, BernelHawkNewFrameCallBack callBack,  
void* pUserData)
```

**[Params]**

streamFlags[in] : The value of the type of stream you want to open.

callBack[in]: A callback function for the sensor streams.

pUserData[in]: A pointer to the user class.

**[Return value]**

0: Success

Other: Failed

**[Remarks]**

“BernelHawkStreamType” is an enumeration of stream types, please refer to the section 3.4.2.

### 3.2.3. stopStreams

**[Description]**

Close streams.

**[Function]**

```
int32_t stopStreams(int streamFlags)
```

**[Params]**

streamFlags[in] : The value of the type of stream you want to close.

**[Return value]**

0 : Success

Other : Failed

**[Remarks]**

“BernelHawkStreamType” is an enumeration of stream types, please refer to the section 3.4.2.

### 3.2.4. getSupportFrameModes

**[Description]**

Get a list of frame modes supported by the data stream.

**[Function]**

```
int32_t getSupportFrameModes(BernelHawkStreamType streamType,  
const BernelHawkStreamFrameMode** pModes, uint32_t* pCount)
```

## Documentation

**[Params]**

streamType[in]: Stream type.  
pModes[out]: A pointer to the frame mode list.  
pCount [out]: The count of the frame mode list.

**[Return value]**

0 : Success  
Other : Failed

**[Remarks]**

BerkelHawkStreamType: Refer to the section 3.4.2.  
BerkelHawkStreamFrameMode: Refer to the section 3.4.5.

### 3.2.5. setFrameMode

**[Description]**

Set frame mode.

**[Function]**

```
int32_t setFrameMode(BerkelHawkStreamType streamType,  
                    BerkelHawkStreamFrameMode *mFrameMode)
```

**[Params]**

streamType[in]: Stream type.  
mFrameMode[in]: A pointer to frame mode.

**[Return value]**

0 : Success  
Other : Failed

**[Remarks]**

BerkelHawkStreamType: Refer to the section 3.4.2.  
BerkelHawkStreamFrameMode: Refer to the section 3.4.5.

### 3.2.6. getCurrentFrameMode

**[Description]**

Get the current frame mode of the stream.

**[Function]**

```
int32_t getCurrentFrameMode(BerkelHawkStreamType streamType,  
                            BerkelHawkStreamFrameMode* frameMode)
```

**[Params]**

streamType[in]: Stream type.  
frameMode[out]: A pointer to the current frame mode.

**[Return value]**

0 : Success  
Othre : Failed

**[Remarks]**

## Documentation

BernelHawkStreamType: Refer to the section 3.4.2.

BernelHawkStreamFrameMode: Refer to the section 3.4.5.

### 3.2.7. readColorFrame

**[Description]**

Read the color frames.

**[Function]**

```
int32_t readColorFrame(BernelHawkFrame* &pFrame, int32_t timeout = 30)
```

**[Params]**

pFrame[out]: A pointer to the frames.

timeout[in]: Set timeout value. If the timeout, the frame will be null.

**[Return value]**

- 0 : Success
- 4: Illegal parameter
- 9: Device disconnect
- 10: Protocol channel Exception
- 11: Read data timeout
- 13: Illegal handle(Failed to open the stream)
- 1: Other error

**[Remarks]**

“BernelHawkFrame” is a class for the sensor frame, please refer to the header file “BernelHawkFrame.h”.

### 3.2.8. readDepthFrame

**[Description]**

Read the depth frames.

**[Function]**

```
int32_t readDepthFrame(BernelHawkFrame* &pFrame, int32_t timeout = 30)
```

**[Params]**

pFrame[out]: A pointer to the frames.

timeout[in]: Set timeout value. If the timeout, the frame will be null.

**[Return value]**

- 0 : Success
- 4: Illegal parameter
- 9: Device disconnect
- 10: Protocol channel Exception
- 11: Read data timeout
- 13: Illegal handle(Failed to open the stream)
- 1: Other error

**[Remarks]**

## Documentation

“BernelHawkFrame” is a class for the sensor frame, please refer to the header file “BernelHawkFrame.h”.

### 3.2.9. readIrFrame

**[Description]**

Read the infrared frames.

**[Function]**

```
int32_t readIrFrame(BernelHawkFrame* &pFrame, int32_t timeout = 30)
```

**[Params]**

pFrame[out]: A pointer to the frames.

timeout[in]: Set timeout value. If the timeout, the frame will be null.

**[Return value]**

- 0 : Success
- 4: Illegal parameter
- 9: Device disconnect
- 10: Protocol channel Exception
- 11: Read data timeout
- 13: Illegal handle(Failed to open the stream)
- 1: Other error

**[Remarks]**

“BernelHawkFrame” is a class for the sensor frame, please refer to the header file “BernelHawkFrame.h”.

### 3.2.10. readLightIrFrame

**[Description]**

Read the infrared speckle frames.

**[Function]**

```
int32_t readLightIrFrame( BernelHawkFrame* &pFrame, int32_t timeout = 30)
```

**[Params]**

pFrame[out]: A pointer to the frames.

timeout[in]: Set timeout value. If the timeout, the frame will be null.

**[Return value]**

- 0 : Success
- 4: Illegal parameter
- 9: Device disconnect
- 10: Protocol channel Exception
- 11: Read data timeout
- 13: Illegal handle(Failed to open the stream)
- 1: Other error

**[Remarks]**

## Documentation

“BernelHawkFrame” is a class for the sensor frame, please refer to the header file “BernelHawkFrame.h”.

### 3.2.11. releaseFrame

**[Description]**

Release the frame handle.

**[Function]**

```
int32_t releaseFrame(BernelHawkFrame* &pFrame)
```

**[Params]**

pFrame[in]: A pointer to the frames.

**[Return value]**

0 : Success

Other : Failed

**[Remarks]**

“BernelHawkFrame” is a class for the sensor frame, please refer to the header file “BernelHawkFrame.h”.

### 3.2.12. startUpgrade

**[Description]**

Upgrade the device and set a callback function for the state of the device.

**[Function]**

```
int32_t startUpgrade(BernelHawkUpgradeProcessCallBack pCallbacks, void* pUserData, const char* pFwFilePath)
```

**[Params]**

pCallbacks[in]: A callback function for the state of the device.

pUserData[in]: A pointer to the user class.

pFwFilePath[in] : The path of the upgrade file

**[Return value]**

0 : Success

Other : Failed

**[Remarks]**

The function “BernelHawkDeviceStatusChangeCallback” is a callback function for device state, please refer to the header file “BernelHawkDefines.h”.

### 3.2.13. convertDepthToPointCloud

**[Description]**

Convert the depth data to point cloud data.

## Documentation

**[Function]**

```
int32_t convertDepthToPointCloud(BernelHawkFrame* pFrame , float factor,  
BernelHawkPoint3D* pPointClouds)
```

**[Params]**

pFrame [in]: The depth frame.

pFactor[in]: If the output point cloud coordinates are in m, input 1000.0, and input 1.0 in mm.

pPointClouds[out]: The point cloud data.

**[Return value]**

0 : Success.

Other : Failed

**[Remarks]**

“BernelHawkPoint3D” is a structure about point cloud data, please refer to the header file “BernelHawkDefines.h”.

### 3.2.14. getVersion

**[Description]**

Get the version.

**[Function]**

```
int32_t getVersion(BernelHawkVersions* versions)
```

**[Params]**

versions[out]: A pointer to the versions.

**[Return value]**

0 : Success.

Other : Failed

**[Remarks]**

“BernelHawkVersions” is a structure about versions, please refer to the header file “BernelHawkDefines.h”.

### 3.2.15. getCurrentDeviceInfo

**[Description]**

Get the information about the currently opened device.

**[Function]**

```
int32_t getCurrentDeviceInfo(BernelHawkDeviceInfo* pDeviceInfo)
```

**[Params]**

pDeviceInfo[out]: A pointer to the current device information.

**[Return value]**

0 : Success

Other : Failed

**[Remarks]**



## Documentation

“BernelHawkDeviceInfo” is a structure about the device information, please refer to the header file “BernelHawkDefines.h”.

### 3.2.16. getCameraIntriscParams

**[Description]**

Get the infrared internal parameter of the camera, which can be used for point cloud data conversion.

**[Function]**

```
int32_t getCameraIntriscParams(BernelHawkCameraIntrinsic *pParams )
```

**[Params]**

pParams [out]: A pointer to the camera infrared internal parameter.

**[Return value]**

0 : Success

Other : Failed

**[Remarks]**

“BernelHawkCameraIntrinsic” is a structure about the camera infrared internal parameter, and the params are calibrated based on the resolution 640\*400. Please refer to the section 3.4.6.

### 3.2.17. getDeviceIntriscParams

**[Description]**

Get the device internal parameter, including camera color internal parameters, infrared internal parameters, translation matrix, rotation matrix and other parameters.

**[Function]**

```
int32_t getDeviceIntriscParams(BernelHawkDeviceIntrinsicParams *pParams )
```

**[Params]**

pParams [out]: A pointer to the device internal parameter.

**[Return value]**

0 : Success

Other : Failed

**[Remarks]**

“BernelHawkDeviceIntrinsicParams” is a structure about the device internal parameter, and the params are calibrated based on the resolution 640\*400. Please refer to the section 3.4.6 and 3.4.7.

### 3.2.18. getNetParams

**[Description]**

## Documentation

Get network camera details, including IP, subnet mask, gateway, etc.

### [Function]

```
int32_t getNetParams(void* pData, uint32_t needDataSize)
```

### [Params]

pData: Network camera details. (See the structure “BeroxelHawkNetParams”).

needDataSize: The size of the structure.

### [Return value]

0 : Success

Other : Failed

### [Remarks]

“BeroxelHawkNetParams” is a structure about the network camera details, please refer to the header file “BeroxelHawkDefines.h”.

## 3.2.19. setStreamMirror

### [Description]

Set image mirroring.

### [Function]

```
int32_t setStreamMirror(bool bNeedMirror)
```

### [Params]

bNeedMirror[in] : true-mirroring, false-non-mirroring

### [Return value]

0 : Success

Other : Failed

## 3.2.20. setRegistrationEnable

### [Description]

Enable or disable the image registration function. The function is disabled by default.

### [Function]

```
int32_t setRegistrationEnable(bool bEnable)
```

### [Params]

bEnable[in]: true : enable registration, false : disable registration.

### [Return value]

0 : Success

Other : Failed

### 3.2.21. setStreamFlagMode

**[Description]**

Set stream flag, including single mode(only open one stream) and mixed streams mode(including mix qvga mode, mix vga mode, mix hd mode). The mix vga mode is enabled by default(The resolution is 640 \* 400). This interface needs to be called before startStream.

**[Function]**

```
int32_t setStreamFlagMode(BernelHawkStreamFlagMode flagMode)
```

**[Params]**

flagMode[in]: Stream mode flag.

**[Return value]**

0 : Success

Other : Failed

### 3.2.22. setSystemClock

**[Description]**

Synchronize the system time of the host and the device.

**[Function]**

```
int32_t setSystemClock()
```

**[Params]**

NULL

**[Return value]**

0 : Success

Other : Failed

### 3.2.23. setNetParams

**[Description]**

Set the ip, gateway, subnet mask and other parameters of the current device.

**[Function]**

```
int32_t setNetParams(void* pData, uint32_t dataSize)
```

**[Params]**

pData: Network camera details. (See the structure "BernelHawkNetParams").

dataSize: The size of the structure.

**[Return value]**

0 : Success

Other : Failed

**[Remarks]**

## Documentation

“BeroxelHawkNetParams” is a structure about the network camera details, please refer to the header file “BeroxelHawkDefines.h”.

### 3.2.24. reconnectNetDevice

**[Description]**

Restart device(only support network device).

**[Function]**

```
int32_t reconnectNetDevice()
```

**[Return value]**

0 : Success

Other : Failed

### 3.2.25. setDenoiseStatus

**[Description]**

Enable or disable the denoise. The function is enable by default.

**[Function]**

```
int32_t setDenoiseStatus(bool bEnable)
```

**[Params]**

bEnable[in]: true : enable denoise, false : disable denoise.

**[Return value]**

0 : Success

Other : Failed

### 3.2.26. openNoiseFilter

**[Description]**

Enable or disable noise filter..

**[Function]**

```
int32_t openNoiseFilter(bool bEnable)
```

**[Params]**

bEnable[in]: true : enable denoise filter, false : disable denoise filter.

**[Return value]**

0 : Success

Other : Failed

### 3.2.27. setTemperatureCompensationStatus

**[Description]**

Enable or disable the temperature compensation. The function is disabled by default.

**[Function]**

```
int32_t setTemperatureCompensationStatus(bool bEnable)
```

**[Params]**

bEnable[in]: true : enable the temperature compensation, false: disable the temperature compensation.

**[Return value]**

0 : Success  
Other : Failed

### 3.2.28. setFrameSync

**[Description]**

Enable or disable the frame synchronization function. The function is enabled by default.

**[Function]**

```
int32_t setFrameSync(bool bEnable)
```

**[Params]**

bEnable[in]: true : enable frame synchronization, false : disable frame synchronization.

**[Return value]**

0 : Success  
Other : Failed

### 3.2.29. enableColorAutoExposure

**[Description]**

Enable color ae function.

**[Function]**

```
int32_t enableColorAutoExposure()
```

**[Params]**

NULL

**[Return value]**

0 : Success  
Other : Failed

### 3.2.30. setColorExposureGain

**[Description]**

Set color frame exposure time and gain.

**[Function]**

```
int32_t setColorExposureGain(uint32_t exposureTime, uint32_t gain)
```

**[Params]**

exposureTime[in]: Color frame exposure time. The parameter is optional: (10000, 20000, 30000).

gain[in]: Color frame gain. The parameter is optional : [100-300].

**[Return value]**

0 : Success

Other : Failed

### 3.2.31. getColorExposureGain

**[Description]**

Get the exposure time, gain and ae state of the color frame.

**[Function]**

```
int32_t getColorExposureGain(uint32_t* exposureTime, uint32_t* gain,  
uint8_t* aeStatus)
```

**[Params]**

exposureTime[out]: The exposure time of the color frame.

gain[out]: The gain of the color frame.

aeStatus: 1: enable, 0 : disable

**[Return value]**

0 : Success

Other : Failed

### 3.2.32. setDepthAESTatus

**[Description]**

Set ae state of the depth frame.

**[Function]**

```
int32_t setDepthAESTatus(bool bEnable)
```

**[Params]**

bEnable[in]: true: enable ae, false : disable ae.

**[Return value]**

0 : Success

Other : Failed

### 3.2.33. getDepthAEStatus

**[Description]**

Get ae state of the depthe frame.

**[Function]**

```
int32_t getDepthAEStatus(uint32_t* value)
```

**[Params]**

value[out]: 1 enable ae, 0 disable ae.

**[Return value]**

0 : Success

Other : Failed

### 3.2.34. setDepthGain

**[Description]**

Set the gain of the depth frame.

**[Function]**

```
int32_t setDepthGain(uint32_t value)
```

**[Params]**

Value[in]: The gain of the depth frame. Selectable range:[1-4].

**[Return value]**

0 : Success

Other : Failed

### 3.2.35. getDepthGain

**[Description]**

Get the gain of the depth frame.

**[Function]**

```
int32_t getDepthGain(uint32_t* value)
```

**[Params]**

Value[in]: The gain of the depth frame.

**[Return value]**

0 : Success

Other : Failed

### 3.2.36. setDepthExposure

**[Description]**

Set the exposure time of the depth frame.

## Documentation

**[Function]**

```
int32_t setDepthExposure(uint32_t value)
```

**[Params]**

Value[in]: The exposure time of the depth frame. Selectable range:[1-43]

**[Return value]**

0 : Success

Other : Failed

### 3.2.37. getDepthExposure

**[Description]**

Get the exposure time of the depth frame.

**[Function]**

```
int32_t getDepthExposure(uint32_t* value)
```

**[Params]**

Value[out]: The exposure time of the depth frame.

**[Return value]**

0 : Success

Other : Failed

### 3.2.38. setDepthElectricCurrent

**[Description]**

Set the electric current value of the depth sensor.

**[Function]**

```
int32_t setDepthElectricCurrent(uint32_t value)
```

**[Params]**

Value[in]: The electric current value of the depth sensor. Selectable range:[200-2000].

**[Return value]**

0 : Success

Other : Failed

### 3.2.39. getDepthElectricCurrent

**[Description]**

Get the electric current value of the depth sensor.

**[Function]**

```
int32_t getDepthElectricCurrent (uint32_t* value)
```

**[Params]**

Value[out]: The electric current value of the depth sensor.



## Documentation

**[Return value]**

0 : Success  
Other : Failed

**3.2.40. setDepthCloseRangeDefaultGainAndExposure****[Description]**

Set the default gain and exposure time for the depth frame, only support the device 100H. (Use the scene up close.)

**[Function]**

```
int32_t setDepthCloseRangeDefaultGainAndExposure()
```

**[Params]**

NULL

**[Return value]**

0 : Success  
Other : Failed

**3.2.41. enableDeviceSlaveMode****[Description]**

Set current device mode(master-slave mode).

**[Function]**

```
int32_t enableDeviceSlaveMode (bool bEnable)
```

**[Params]**

bEnable[in]: true: enable slave mode, false: disable slave mode.

**[Return value]**

0 : Success  
Other : Failed

**3.2.42. getDeviceMasterSlaveMode****[Description]**

Get current device mode. (master-slave mode)

**[Function]**

```
int32_t getDeviceMasterSlaveMode (uint32_t* value)
```

**[Params]**

value[out]: 1:Slave mode 0: master mode

**[Return value]**

0 : Success  
Other : Failed

### 3.2.43. setColorQuality

**[Description]**

Set color quality.

**[Function]**

```
int32_t setColorQuality(uint32_t nValue)
```

**[Params]**

nValue[in]: color quality. Range[20-80].

**[Return value]**

0 : Success

Other : Failed

### 3.2.44. getColorQuality

**[Description]**

Get color quality.

**[Function]**

```
int32_t getColorQuality(uint32_t* nValue)
```

**[Params]**

nValue[out]: color quality.

**[Return value]**

0 : Success

Other : Failed

### 3.2.45. setDepthConfidence

**[Description]**

Set depth confidence.

**[Function]**

```
int32_t setDepthConfidence(uint32_t nValue)
```

**[Params]**

nValue[in]: depth confidence. Range[1-5].

**[Return value]**

0 : Success

Other : Failed

### 3.2.46. getDepthConfidence

**[Description]**

## Documentation

Get depth confidence.

### [Function]

```
int32_t getDepthConfidence(uint32_t* nValue)
```

### [Params]

nValue[out]: depth confidence.

### [Return value]

0 : Success

Other : Failed

## 3.2.47. enableHightFpsMode

### [Description]

Set depth high fps mode (45 fps for vga or qvga).

### [Function]

```
int32_t enableHightFpsMode(bool bEnable)
```

### [Params]

bEnable[in]: true: enable high fps mode, false: disable high fps mode.

### [Return value]

0 : Success

Other : Failed

## 3.2.48. setSpatialDenoiseStatus

### [Description]

Set spatial denoise status for depth.

### [Function]

```
int32_t setSpatialDenoiseStatus(bool bEnable)
```

### [Params]

bEnable[in]: true: enable spatial denoise, false: disable spatial denoise.

### [Return value]

0 : Success

Other : Failed

## 3.2.49. setTemporalDenoiseStatus

### [Description]

## Documentation

Set temporal denoise status for depth.

### [Function]

```
int32_t setTemporalDenoiseStatus(bool bEnable)
```

### [Params]

bEnable[in]: true: enable temporal denoise, false: disable temporal denoise.

### [Return value]

0 : Success

Other : Failed

## 3.2.50. enableDecodeColorData

### [Description]

Set rgb data format.

### [Function]

```
int32_t enableDecodeColorData(bool bEnable)
```

### [Params]

bEnable[in]: true: RGB888 data, false: jpeg data.

### [Return value]

0 : Success

Other : Failed

## 3.3. BernelHawkFrame Module Description

This module is mainly used to get the details of the frame, refer to the Header file "BernelHawkFrame.h".

The functions are described as follows:

Function	Description
BernelHawkPixelType getPixelType()	Get the pixformat of the image frame.
BernelHawkStreamType getStreamType()	Get the stream type of the image frame.
uint32_t getFrameIndex()	Get the frame index of the image frame.
uint64_t getTimeStamp()	Get the timestamp of the image frame.
uint32_t getFPS()	Get the frame rate of the image frame.
uint32_t getWidth()	Get the width of the image frame.
uint32_t getHeight()	Get the height of the image frame.
void* getData()	Get the frame data.
uint32_t getDataSize()	Get the size of the image frame.

### 3.4. BernelHawkDefines Module Description

This module is mainly used to define the relevant enumerations and struct information required by the SDK API, refer to the BernelHawkDefines.h header file.

#### 3.4.1. BernelHawkPixelType Enumeration Description

This enumeration represents the pixel format and is described as follows:

Definition	Description
BERXEL_HAWK_PIXEL_TYPE_IMAGE_RGB24	The pixel format of the color frame. 3 bytes per pixel. (RGB888)
BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D	The pixel format of the depth frame. 2 bytes per pixel. The first 12 bits represent integer bits. The last 4 digits represent decimal places.
BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D	The pixel format of the depth frame. 2 bytes per pixel. The first 13 bits represent integer bits. The last 3 digits represent decimal places.
BERXEL_HAWK_PIXEL_TYPE_IR_16BIT	The pixel format of the infrared frame. 2 bytes per pixel
BERXEL_HAWK_PIXEL_INVALID_TYPE	Unsupported pixel format

#### 3.4.2. BernelHawkStreamType Enumeration Description

This enumeration represents the stream type, as follows:

Definition	Description
BERXEL_HAWK_COLOR_STREAM	The stream type of the color frame.
BERXEL_HAWK_DEPTH_STREAM	The stream type of the depth frame.
BERXEL_HAWK_IR_STREAM	The stream type of the ir frame.
BERXEL_HAWK_LIGHT_IR_STREAM	The stream type of the light ir frame.
BERXEL_HAWK_INVALID_STREAM	Unsupported stream types.

#### 3.4.3. BernelHawkStreamFlagMode Enumeration Description

This enumeration represents a stream mode type, in SINGULAR mode, only switches for a

## Documentation

single stream are supported. In MIX mode, a hybrid switch for multiple data streams is supported, as follows:

Definition	Description
BERXEL_HAWK_SINGULAR_STREAM_FLAG_MODE	The data stream is in single mode, and only one type of data stream can be opened.
BERXEL_HAWK_MIX_STREAM_FLAG_MODE	The data stream is a mixed-stream VGA mode, which supports opening multiple data streams at the same time
BERXEL_HAWK_MIX_HD_STREAM_FLAG_MODE	The data stream is in mixed streaming HD mode, which supports opening multiple data streams at the same time
BERXEL_HAWK_MIX_QVGA_STREAM_FLAG_MODE	The data stream is a mixed-stream QVGA mode, which supports opening multiple data streams at the same time

The horizontal device supports the following resolutions:

Stream Mode	Color Resolution	Depth Resolution	IR Resolution
BERXEL_HAWK_SINGULAR_STREAM_FLAG_MODE	1920*1080@30fps 640*400@30fps	320*200@5fps 320*200@10fps 320*200@15fps 320*200@20fps 320*200@25fps 320*200@30fps 640*400@5fps 640*400@10fps 640*400@15fps 640*400@20fps 640*400@25fps 640*400@30fps 1280*800@10fps	640*400@30fps
BERXEL_HAWK_MIX_STREAM_FLAG_MODE	640*400@30fps	640*400@5fps 640*400@10fps 640*400@15fps 640*400@20fps 640*400@25fps 640*400@30fps	640*400@30fps
BERXEL_HAWK_MIX_HD_STR	1280*800@10fps	1280*800@10fps	Not supported

## Documentation

EAM_FLAG_MODE			
BERXEL_HAWK_MIX_QVGA_S TREAM_FLAG_MODE	320*240@30fps	320*200@5fps 320*200@10fps 320*200@15fps 320*200@20fps 320*200@25fps 320*200@30fps	Not supported

### 3.4.4. BernelHawkDeviceInfo Structure Description

This structure represents the device information, as follows:

Definition	Data Type	Description
vendorId	uint16_t	Vendor id
productId	uint16_t	Product id
devBus	uint32_t	Usb-Bus
devPort	uint32_t	Usb-Port
deviceNum	uint32_t	Usb-Device number
deviceType	uint32_t	Device Type
serialNumber	char[]	Serial Number
deviceAddress	char[]	Device Address

### 3.4.5. BernelHawkStreamFrameMode Structure Description

This struct represents the frame mode and is required in setting and get the FrameMode, as follows:

Definition	Data Type	Description
pixelType	BernelHawkPixelFormat	Pixel format, refer to the section 3.4.1
resolutionX	int16_t	width
resolutionY	int16_t	height
framerate	int8_t	fps

### 3.4.6. BernelHawkCameraIntrinsic Structure Description

This structure represents the camera parameters, which are used when converting point cloud data, as follows:

Definition	Data Type	Description
fxParam	float	X-direction focal length

## Documentation

fyParam	float	Y-direction focal length
cxParam	float	X-direction primary optical pivot point position
cyParam	float	Y-direction primary optical pivot point position
k1Param	float	Radial distortion K1
k2Param	float	Radial distortion K2
p1Param	float	Tangential distortion p1
p2Param	float	Tangential distortion p2
k3Param	float	Radial distortion K3

### 3.4.7. BixelHawkDeviceIntrinsicParams Structure Description

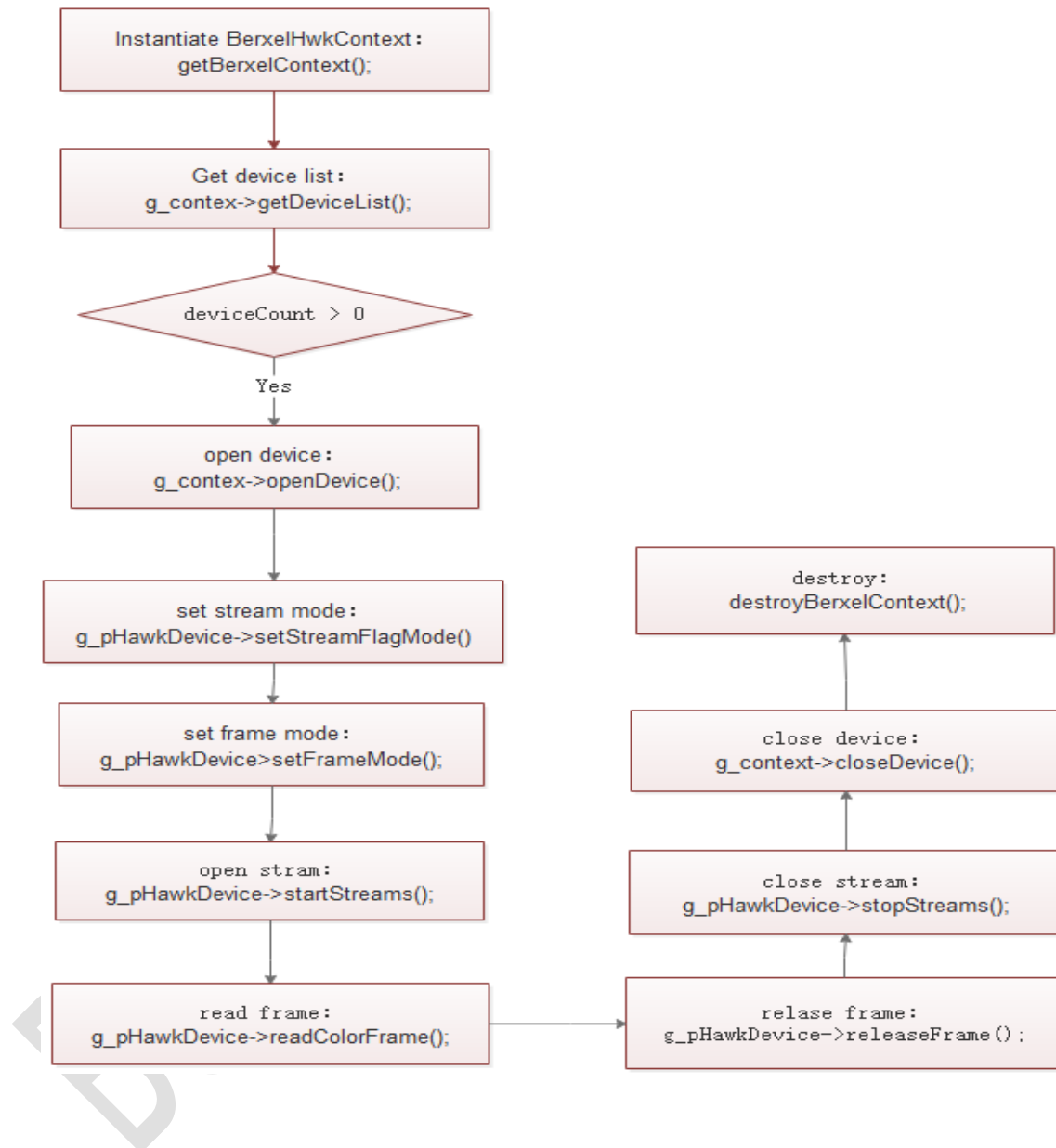
This structure represents a device internal parameter, For details, see 3.4.6.

Definition	Data Type	Description
colorIntrinsicParams	int8_t[]	Color internal parameter, sizeof (float) * 9
irIntrinsicParams	int8_t[]	Ir internal parameter, sizeof (float) * 9
liteIrIntrinsicParams	int8_t[]	Light ir internal parameter, sizeof (float) * 9
rotateIntrinsicParams	int8_t[]	Rotate the matrix, sizeof (float) * 9
translationIntrinsicParams	int8_t[]	Translation matrix, sizeof (float) * 3



## 4. SDK Guidelines

### 4.1. Call SDK API Flowchart



### 4.2. Get Color Frame

Refer to HawkColor example program in SDK.

### 4.3. Get Depth Frame

Refer to HawkDepth example program in SDK

### 4.4. Get Color and Depth Mix Frame

Refer to HawkMixColorDepth example program in SDK.

### 4.5. Get Version

The version number structure is defined in the BernelHawkDefinitions. h header file as BernelHawkVersions, including SDK version number, firmware version number, and hardware version number. The sample code is as follows:

```
berxel::BernelHawkVersions tempVersions ;  
g_pHawkDevice->getVersion(&tempVersions);
```

### 4.6. Get Current Device Info

The device info structure is defined in the BernelHawkDefinitions. h header file as BernelHawkDeviceInfo, including SN, vendorId, productId, deviceNum, serialNumber, and deviceAddress. The sample code is as follows:

```
berxel::BernelHawkDeviceInfo tempCurInfo;  
g_pHawkDevice->getCurrentDeviceInfo(&tempCurInfo);
```

### 4.7. Depth Raw Data Converted To The Distance Value

The depth raw data has two data formats, BERXEL\_HAWK\_PIXEL\_TYPE\_DEP\_16BIT\_12I\_4D and BERXEL\_HAWK\_PIXEL\_TYPE\_DEP\_16BIT\_13I\_3D.

The two formats are distinguished according to the getPixelFormat interface in the BernelHawkFrame.

#### 4.7.1. BERNEL\_HAWK\_PIXEL\_TYPE\_DEP\_16BIT\_12I\_4D

The depth raw data has 16 bits. The upper 12 bits are integer parts and the lower 4 bits are decimal parts, The conversion steps are as follows:

- ① Get raw data: `uint16_t depthOri = pDepth[i];` (pDepth is depth data pointer)
- ② Get integer part: `float depthFront = depthOri >> 4;`
- ③ Get decimal part: `float depthTail = (depthOri & 0x000f)/16;`
- ④ Get depth value: `float depth = depthFront + depthTail;`

#### 4.7.2. BERNEL\_HAWK\_PIXEL\_TYPE\_DEP\_16BIT\_13I\_3D

The depth raw data has 16 bits. The upper 13 bits are integer parts and the lower 3 bits are decimal parts, The conversion steps are as follows:

- ① Get raw data: `uint16_t depthOri = pDepth[i];` (pDepth is depth data pointer)
- ② Get integer part: `float depthFront = depthOri >> 3;`
- ③ Get decimal part: `float depthTail = (depthOri & 0x0007)/8;`
- ④ Get depth value: `float depth = depthFront + depthTail;`

### 4.8. Depth Raw Data Converted To Point Cloud

The sample code is as follows:

```
//Step1: get depth data
berxel::BernelHawkFrame *pHawkFrame = NULL;
g_pHawkDevice->readDepthFrame(pHawkFrame, 30);

//Step2: call convertDepthToPointCloud function
static berxel::BernelHawkPoint3D m_3dPoints[400 * 640];
g_pHawkDevice->convertDepthToPointCloud(pHawkFrame, 1000.0, m_3dPoints);
```

Notes: please refer to section 3.2.13 for the `convertDepthToPointCloud` function

### 4.9. Set Device Status Monitoring

The `onDeviceStatusChange` function can be used to monitor the disconnection and connection status of the device, The sample code is as follows:

```
void _stdcall Test::onDeviceStatusChange(const char* deviceAddr,
```

## Documentation

```
berxel::BernelHawkDeviceStatus deviceState, void* pUserData)
{
    if (NULL != pUserData)
    {
        Test* pTest = (Test*)pUserData;
        if (NULL != pTest)
        {
            pTest->processDeviceStatusChange(deviceAddr, deviceState);
        }
    }
}

int32_t Test::processDeviceStatusChange(const char* deviceAddr,
berxel::BernelHawkDeviceStatus deviceState) {

    switch(deviceState)
    {
        case berxel::BERXEL_HAWK_DEVICE_DISCONNECT:
            break;
        case berxel::BERXEL_HAWK_DEVICE_CONNECT:
            break;
        default:
            break;
    }
    return 1;
}

int32_t Test::init ()
{
    m_context->setDeviceStateCallback(onDeviceStatusChange, this);
}
```

## 4.10. Get Device Intrinsic Params

We can obtain the camera parameters as follows:

```
berxel::BernelHawkDeviceIntrinsicParams intrinsicParams;
memset(&intrinsicParams, 0x00, sizeof(berxel::BernelHawkDeviceIntrinsicParams));
g_pHawkDevice->getDeviceIntriscParams(&intrinsicParams);
```

## Documentation

```
berxel::BernelHawkCameraIntrinsic colorParams;  
berxel::BernelHawkCameraIntrinsic depthParams;  
float rotateParams[9] = { 0x00 };  
float transParams[3] = { 0x00 };  
  
memset(&colorParams, 0x00, sizeof(berxel::BernelHawkCameraIntrinsic));  
memset(&depthParams, 0x00, sizeof(berxel::BernelHawkCameraIntrinsic));  
  
//color intrinsic params  
memcpy((uint8_t*)&colorParams, intrinsicParams.colorIntrinsicParams,  
sizeof(berxel::BernelHawkCameraIntrinsic));  
  
//depth intrinsic params  
memcpy((uint8_t*)&depthParams, intrinsicParams.liteIrIntrinsicParams,  
sizeof(berxel::BernelHawkCameraIntrinsic));  
  
//rotation matrix params  
memcpy(rotateParams, intrinsicParams.rotateIntrinsicParams, sizeof(rotateParams));  
  
//translation matrix  
memcpy(transParams, intrinsicParams.translationIntrinsicParams,  
sizeof(transParams));
```

## 4.11. SDK Error Code

Error code	Description
0	succeed
-1	failed
-2	not init
-3	device not opened
-4	illegal params
-5	illegal operation

## Documentation

-6	Failed to send standard command
-7	Failed to send private command
-8	unsupported property operation
-9	abnormal equipment channel
-10	device disconnected
-11	read frame timeout
-12	driver not supported
-13	stream not opened