Berxel SDK Developer Documentation

Revision History

Version	Sections Modified	Summary of Changes	Date	Author
V1.0	-	Initial draft completed	2021.07.28	Allen
V1.1		1. Added interface for retrieving camera intrinsic parameters 2. Added interface for enabling image mirroring 3. Added interface for retrieving point cloud data 4. Added description for converting depth frames to real-world distance values	2021.08.02	Allen
V1.2		Modified depth-to-point cloud conversion API	2021.08.27	Allen
V1.3		1. Added interface to enable or disable registration (depth-color alignment) 2. Added interface to configure stream mode settings	2021.09.02	Allen
V1.4		Added API for synchronizing system time to the device	2021.09.14	Allen
V2.0.16		Added documentation related to the C# SDK	2021.10.18	Allen
V2.0.18		1. Modified parameters of closeDevice interface 2. Modified parameters of startUpgrade interface	2021.10.19	Allen
V2.0.21		1. Updated enum definitions in section 3.4.1 2. Added example code for device status monitoring (section 4.10) 3. Added ROS SDK related notes	2021.11.02	Allen

V2.0.22		1. Added interface for configuring depth image exposure 2. Added interface for configuring depth image gain 3. Added interface for enabling Auto Exposure (AE) for depth stream	2022.05.12	Sun
V2.0.23		1. Added interfaces for querying depth image exposure/gain/current/AE status 2. Added interfaces for querying color image exposure/gain/AE status 3. Added interfaces for configuring and querying GPIO status	2022.07.12	Sun
V2.0.24		1. Added setStreamStatus API 2. Added setBindCpuCore API	2022.11.24	Sun
V2.0.25		Added setDepthRemoteMode API for managing long-range depth output behavior	2023.02.22	Allen
V2.0.26		Added APIs for adjusting the range limits of exposure time and gain under AE mode	2023.03.22	Sun

Table of Contents

1. Overview **9**

1.1. SDK Introduction 9

1.2. SDK Compatibility 9

2. SDK Package Structure 9

2.1. SDK Components 9

2.2. SDK Samples 9

2.2.1. Sample Descriptions 9

2.2.2. Sample UI Display Instructions 10

2.3. SDK Integration 10

2.3.1. Windows SDK API Integration Steps 10

2.3.2. Linux SDK API Integration Steps 10

3. SDK API Reference 10

3.1. BerxelHawkContext Module Description 10

3.1.1. get Berxel Context 10

3.1.2. destroy Berxel Context 11

3.1.3. get Device List 11

3.1.4. open Device 11

3.1.5. close Device 12

3.1.6. set Device State Callback 12

3.2. BerxelHawkDevice Module Description 13

3.2.1. start Streams 13

3.2.2. start Streams 13

3.2.3. stop Streams 13

3.2.4. get Support Frame Modes 14

3.2.5. set Frame Mode 14

3.2.6. get Current Frame Mode 15

3.2.7. read Color Frame 15
3.2.8. read Depth Frame 16
3.2.9. read Ir Frame 16
3.2.10. read Light Ir Frame 17
3.2.11. release Frame 17
3.2.12. start Upgrade 17
3.2.13. convert Depth To Point Cloud 18
3.2.14. get Version 18
3.2.15. get Current Device Info 19
3.2.16. get Camera Intrisc Params 19
3.2.17. get Device Intrisc Params 19
3.2.18. get Net Params 20
3.2.19. set Stream Mirror 20
3.2.20. set Registration Enable 20
3.2.21. set Stream Flag Mode 21
3.2.22. set System Clock 21
3.2.23. set Net Params 21
3.2.24. reconnect Net Device 22
3.2.25. set Denoise Status 22
3.2.26. set Temperature Compensation Enable 22
3.2.27. set Frame Sync 23
3.2.28. enablen Color Auto Exposure 23
3.2.29. set Color Exposure Gain 23
3.2.30. get Color Exposure Gain 24
3.2.31. set Depth AE Status 24
3.2.32. get Depth AE Status 25
3.2.33. set Depth Gain 25
3.2.34. get Depth Gain 25
3.2.35. set Depth Exposure 26
3.2.36. get Depth Exposure 26

3.2.37. set Depth Electric Current 26
3.2.38. get Depth Electric Current 26
3.2.39. set Depth Close Range Default Gain And Exposure 27
3.2.40. enable Device Slave Mode 27
3.2.41. get Device Master Slave Mode 27
3.2.42. set User Gpio Ctrl 28
3.2.43. get User Gpio Ctrl 28
3.2.44. set Stream Status 29
3.2.45. set Bind Cpu Core 29
3.2.46. set Depth Remote Mode 29
3.2.47. set Depth Confidence 30
3.2.48. get Depth Confidence 30
3.2.49. set Edge Optimization Status 30
3.2.50. enable Hight Fps Mode 31
3.2.51. set Depth AE Gain Range 31
3.2.52. get Depth AE Gain Range 31
3.2.53. set Depth AE Exposure Range 32
3.2.54. get Depth AE Exposure Range 32
3.2.55. set Temperature Params File Path 32
3.2.56. set Device TransferMode 33
3.2.57. set Temporal Denoise Status 33
3.2.58. set Spatial Denoise Status 34
3.2.59. set Sync Host Time 34
3.2.60. device Support HwTemp CompenSation 34
3.2.61. reboot Device 35
3.2.62. get Device Log Size 35
3.2.63. get Device Log 35
3.2.64. open Noise Filter 36
3.2.65. set Max Depth Value 36
3.2.66. set Filter Ground Value 36

3.2.67. set Depth Optimization Status 37
3.3. BerxelHawkFrame Module Description 37
3.4. BerxelHawkDefine Module Description 38
3.4.1. BerxelHawkPixelType Enumeration 38
3.4.2. BerxelHawkStreamType Enumeration 38
3.4.3. BerxelHawkStreamFlagMode Enumeration 38
3.4.4. BerxelHawkDeviceInfo Structure Description 41
3.4.5. BerxelHawkStreamFrameMode Structure Description 41
3.4.6. BerxelHawkCameraIntrinsic Structure Description 41
3.4.7. BerxelHawkDeviceIntrinsicParams Structure Description 42
3.4.8. BerxelHawkColorGainTable Description 42
4. SDK Development Guide 44
4.1. SDK API Call Flowchart 44
4.2. Retrieving Color Frames 44
4.3. Retrieving Infrared Frames 45
4.4. Retrieving Infrared Frames 45
4.5. Retrieving Mixed Color + Depth Frames 45
4.6. Retrieving Version Information 45
4.7. Retrieving Current Device Information 45
4.8. Converting Depth Image Data to Depth Values 45
4.8.1. BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D 45
4.8.2. BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D 46
4.9. Converting Depth Image Data to Point Cloud Data 46
4.10. Setting Device Status Listener 46
4.11. Retrieving Intrinsic Parameters 48
4.12. SDK Common Error Codes 48
5. C#SDK Description 49

5.1. C#SDK Overview 49

5.2. C# SDK Development Instructions 49

6. Ros SDK Description 50

6.1. Ros SDK Overview 50

6.2. ROS SDK Development Instructions 50

1. Overview

1.1. SDK Introduction

Berxel SDK is a software development kit based on Berxel 3D cameras. It is applicable in application scenarios requiring 3D imaging in fields such as industrial control and consumer electronics. The SDK supports Android, Windows, Linux, and ROS platforms.

1.2. SDK Compatibility

- Windows 7 / Windows 10 (x86 / x64)
- Ubuntu 14 or later, ARM and x86 / x64 platforms
- USB 2.0, 2A current
- 4 GB RAM or more
- CPU clock speed 2.2 GHz or higher

2. SDK Package Structure

2.1. SDK Components

The SDK installation package is organized into the following modules:

Module Name	Description
Document	SDK documentation
Driver	DFU mode driver, mainly for firmware upgrade
Include	SDK header files
Lib	SDK dynamic/static libraries. Required for API access (BerxelHawk.lib)
Samples	Sample source code, built with Visual Studio 2010
Thirdparty	Third-party libraries required for sample GUI rendering
Tools	Berxel SDK demo tool
C#	C# SDK interface and demo

2.2. SDK Samples

2.2.1. Sample Descriptions

Sample Name	Description

HawkColor	Demonstrates color frame acquisition
HawkDepth	Demonstrates color frame acquisition
HawkIr	Demonstrates infrared frame acquisition
HawkMixColorDepth	Demonstrates acquiring mixed stream (color + depth) at 640×400 resolution
HawkMixHDColorDepth	Demonstrates acquiring mixed stream (color + depth) at 1280×800 resolution
HawkMixColorDepthMatting	Demonstrates registration feature

2.2.2. Sample UI Display Instructions

Samples use OpenGLfor UI display. Notes for platform-specific GL setup:

- Windows: Requires FreeGLUT library. Header and library files are located in Thirdparty/freeglut.
- Ubuntu: Install FreeGLUT using: sudo apt-get install freeglut3-dev

2.3. SDK Integration

2.3.1. Windows SDK API Integration Steps

- Link BerxelHawk.lib from the lib directory
- Include all header files from the Include directory
- Copy all files from lib to your applications output directory

2.3.2. Linux SDK API Integration Steps

- Link BerxelHawk.so from the libs directory in your Makefile
- Add the following environment variable: export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/your/libs/path

3. SDK API Reference

3.1. BerxelHawkContext Module Description

The BerxelHawkContext module is mainly responsible for managing the device list and handling device opening and closing. (Refer to BerxelHawkContext.h)

3.1.1. get Berxel Context

[Description]

Get an instance of the BerxelHawkContext object.

[Function]

static BerxelHawkContext* getBerxelContext()

static BerxelHawkContext* getBerxelContext(const char* strLogPath)

[Params]

strLogPath[in]: Log file output path for the SDK

[Return value]

BerxelHawkContext*: Success

NULL: Failure

3.1.2. destroy Berxel Context

[Description]

Destroy the BerxelHawkContext instance.

[Function]

static void destroyBerxelContext(BerxelHawkContext* &pBerxeContext)

[Params]

pBerxeContext [in]: Pointer to BerxelHawkContext

[Return value]

None

3.1.3. get Device List

[Description]

Retrieve a list of connected devices.

[Function]

int32_t getDeviceList(BerxelHawkDeviceInfo** pDeviceList, uint32_t* pDeviceCount)

[Params]

pDeviceList [out]: Pointer to device list

pDeviceCount [out]: Number of devices found

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkDeviceInfo is defined in BerxelHawkDefines.h

3.1.4. open Device

[Description]

Open a specific device.

[Function]

BerxelHawkDevice* openDevice (const BerxelHawkDeviceInfo deviceInfo)

[Params]

deviceInfo [in]: Target device info (from getDeviceList)

[Return value]

BerxelHawkDevice*: Success

NULL: Failure

[Remarks]

BerxelHawkDevice is defined in BerxelHawkDevice.h

BerxelHawkDeviceInfo is defined in BerxelHawkDefines.h

3.1.5. close Device

[Description]

Close an opened device.

[Function]

int32_t closeDevice((BerxelHawkDevice* hawkDevice)

[Params]

hawkDevice[in]: Handle of the opened device

[Return value]

0: Success

Non-zero: Failure

3.1.6. set Device State Callback

[Description]

Set a device state change callback (connect/disconnect notification).

[Function]

int32_t setDeviceStateCallback(BerxelHawkDeviceStatusChangeCallback callback,void* pUserData)

[Params]

Callback[in]: Function pointer for state change notification

pUserData[in]: Function pointer for state change notification

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkDeviceStatusChangeCallback is defined in BerxelHawkDefines.h

3.2. BerxelHawkDevice Module Description

The BerxelHawkDevice module manages device-level operations, including starting/stopping streams, reading frames, and querying device information. (Refer to BerxelHawkDevice.h)

3.2.1. start Streams

[Description]

Starts the specified data stream(s).

[Function]

int32_t startStreams(int streamFlags)

[Params]

streamFlags [in]: Bitmask specifying stream types to start

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkStreamType is defined in section 3.4.2.

3.2.2. start Streams

[Description]

Starts the specified data stream and register a callback for new frame arrival.

[Function]

int32_t startStreams(int streamFlags, BerxelHawkNewFrameCallBack callBack, void* pUserData)

[Params]

streamFlags [in]: Stream types to be started

callback [in]: Callback function triggered on new frame arrival

pUserData [in]: Pointer to user-defined data. This value will be passed as an input argument to the BerxelHawkNewFrameCallBack function when the callback is invoked.

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkStreamType is defined as an enum. See Section 3.4.2.

3.2.3. stop Streams

[Description]

Stop the specified data streams.

[Function]

int32_t stopStreams(int streamFlags)

[Params]

streamFlags [in]: Stream types to stop, defined in BerxelHawkStreamType

[Return value]

0: Success

Non-zero: Failure

[Remarks]

See BerxelHawkStreamType in Section 3.4.2.

3.2.4. get Support Frame Modes

[Description]

Get the list of supported frame modes for the specified stream.

[Function]

int32_t getSupportFrameModes(BerxelHawkStreamType streamType, const BerxelHawkStreamFrameMode** pModes, uint32_t* pCount)

[Params]

streamType [in]: Stream type

pModes [out]: Pointer to the list of frame modes

pCount [out]: Number of frame modes

[Return value]

0: Success

Non-zero: Failure

[Remarks]

See BerxelHawkStreamType in Section 3.4.2

See BerxelHawkStreamFrameMode in Section 3.4.5

3.2.5. set Frame Mode

[Description]

Set the frame mode for the specified stream.

[Function]

int32_t setFrameMode(BerxelHawkStreamType streamType, BerxelHawkStreamFrameMode *mFrameMode)

[Params]

streamType [in]: Stream type

mFrameMode [in]: Frame mode pointer

[Return value]

0: Success

Non-zero: Failure

[Remarks]

See BerxelHawkStreamType in Section 3.4.2

See BerxelHawkStreamFrameMode in Section 3.4.5

3.2.6. get Current Frame Mode

[Description]

Retrieves the current frame mode of a stream.

[Function]

int32_t getCurrentFrameMode(BerxelHawkStreamType streamType, BerxelHawkStreamFrameMode* frameMode)

[Params]

streamType [in]: Stream type

frameMode [out]: Pointer to receive current frame mode

[Return value]

0: Success

Non-zero: Failure

[Remarks]

See BerxelHawkStreamType in Section 3.4.2

See BerxelHawkStreamFrameMode in Section 3.4.5

3.2.7. read Color Frame

[Description]

Retrieves a color frame.

[Function]

int32_t readColorFrame(BerxelHawkFrame* &pFrame, int32_t timeout = 30)

[Params]

pFrame[out]: Pointer to frame object

timeout[in]: Timeout duration; if timed out, pFrame will be NULL

[Return value]

0: Operation successful

-4: Invalid argument

-9: Device disconnected

-10: Protocol error

-11: Frame acquisition timeout

-13: Invalid stream handle (stream not started)

-1: Unknown error

[Remarks]

BerxelHawkFrame type is defined in BerxelHawkFrame.h.

3.2.8. read Depth Frame

[Description]

Retrieves a depth frame.

[Function]

int32_t readDepthFrame(BerxelHawkFrame* &pFrame, int32_t timeout = 30)

[Params]

pFrame [out]: Pointer to frame object

timeout [in]: Timeout duration; if timed out, pFrame will be NULL

[Return value]

0: Success
-4: Invalid parameter
-9: Device disconnected
-10: Protocol channel error
-11: Read timeout
-13: Invalid stream handle (stream not started successfully)
-1: Other error

[Remarks]

BerxelHawkFrame is defined in BerxelHawkFrame.h.

3.2.9. read Ir Frame

[Description]

Retrieves a flood infrared frame.

[Function]

int32_t readIrFrame(BerxelHawkFrame* &pFrame, int32_t timeout = 30)

[Params]

pFrame [out]: Pointer to frame object

timeout [in]: Timeout duration; if timed out, pFrame will be NULL

[Return value]

0: Success
-4: Invalid parameter
-9: Device disconnected
-10: Protocol channel error
-11: Read timeout
-13: Invalid stream handle (stream not started successfully)
-1: Other error

[Remarks]

BerxelHawkFrame is defined in BerxelHawkFrame.h.

3.2.10. read Light Ir Frame

[Description]

Retrieves a point infrared frame.

[Function]

int32_t readLightIrFrame(BerxelHawkFrame* &pSteamFrame, int32_t timeout = 30)

[Params]

pFrame [out]: Pointer to frame object

timeout [in]: Timeout duration; if timed out, pFrame will be NULL

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkFrame is defined in BerxelHawkFrame.h.

3.2.11. release Frame

[Description]

Releases a retrieved frame.

[Function]

int32_t releaseFrame(BerxelHawkFrame* &pHawFrame)

[Params]

pHawkFrame [in]: Pointer to the frame to be released

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkFrame is defined in BerxelHawkFrame.h.

3.2.12. start Upgrade

[Description]

Performs firmware upgrade and sets upgrade status callback.

[Function]

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

[Params]

pCallbacks [in]: Callback function for upgrade status

pUserData [in]: User-defined pointer, passed as input to the callback

pFwFilePath [in]: Path to the firmware file

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkUpgradeProcessCallBack is defined in BerxelHawkDefines.h.

3.2.13. convert Depth To Point Cloud

[Description]

Converts depth data into 3D point cloud coordinates.

[Function]

int32_t convertDepthToPointCloud(BerxelHawkFrame* pFrame, float factor, BerxelHawkPoint3D* pPointClouds, bool bPointCloudWall = false)

[Params]

pFrame [in]: Input depth frame

factor [in]: Unit scale (e.g., 1000.0 for meters, 1.0 for millimeters)

pPointClouds [out]: Output point cloud array

bPointCloudWall [in]: true to remove wall points, false to retain

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkPoint3D is defined in BerxelHawkDefines.h.

3.2.14. get Version

[Description]

Retrieves version information.

[Function]

int32_t getVersion(BerxelHawkVersions* versions)

[Params]

versions [out]: Pointer to receive version data

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkVersions is defined in BerxelHawkDefines.h.

3.2.15. get Current Device Info

[Description]

Retrieves the information of the currently opened device.

[Function]

int32_t getCurrentDeviceInfo(BerxelHawkDeviceInfo* pDeviceInfo)

[Params]

pDeviceInfo [out]: Pointer to the structure receiving device information

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkDeviceInfo is defined in BerxelHawkDefines.h.

3.2.16. get Camera Intrisc Params

[Description]

Retrieves the infrared camera intrinsic parameters, used for point cloud generation.

[Function]

int32_t getCameraIntriscParams(BerxelHawkCameraIntrinsic *pParams)

[Params]

pParams [out]: Pointer to the structure receiving camera intrinsics

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkCameraIntrinsic is defined in section 3.4.6. Parameters are calibrated for 400×640 (or 640×400) resolution.

3.2.17. get Device Intrisc Params

[Description]

Retrieves device-level intrinsic and extrinsic parameters, including color/IR camera intrinsics and rotation/translation matrices.

[Function]

int32_t getDeviceIntriscParams(BerxelHawkDeviceIntrinsicParams *pParams)

[Params]

pParams [out]: Pointer to the structure receiving all intrinsics

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkDeviceIntrinsicParams is defined in section 3.4.7. Parameters are calibrated for 800×1280 (or 1280×800) resolution.

3.2.18. get Net Params

[Description]

Retrieves network-related parameters from the camera, including IP address, subnet mask, and gateway.

[Function]

int32_t getNetParams(void* pData, uint32_t needDataSize)

[Params]

pData [out]: Pointer to BerxelHawkNetParams structure receiving network info

needDataSize [in]: Size of the network parameter structure

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkNetParams is defined in BerxelHawkDefines.h.

3.2.19. set Stream Mirror

[Description]

Sets the mirror mode for image streams. By default, mirroring is disabled.

[Function]

int32_t setStreamMirror(bool bNeedMirror)

[Params]

bNeedMirror [in]:

true enables mirroring;

false disables mirroring

[Return value]

0: Success

Non-zero: Failure

3.2.20. set Registration Enable

[Description]

Enables or disables depth-to-color registration. Registration is disabled by default.

[Function]

int32_t setRegistrationEnable(bool bEnable)

[Params]

bEnable [in]:

true to enable registration;

false to disable registration

[Return value]

0: Success

Non-zero: Failure

3.2.21. set Stream Flag Mode

[Description]

Sets the stream flag mode, including single and mixed stream modes (QVGA, VGA, HD). Default mode is VGA mixed stream (400×640 or 640×400). This function must be called before stream startup.

[Function]

int32_t setStreamFlagMode(BerxelHawkStreamFlagMode flagMode)

[Params]

flagMode [in]: Stream mode flag value

[Return value]

0: Success

Non-zero: Failure

3.2.22. set System Clock

[Description]

Synchronizes the host system time to the camera. Should be called before starting the stream to ensure frame timestamps match host clock.

[Function]

int32_t setSystemClock()

[Params]

NULL

[Return value]

0: Success

Non-zero: Failure

3.2.23. set Net Params

[Description]

Sets network parameters for the current device, including IP address, subnet mask, and gateway.

[Function]

int32_t setNetParams(void* pData, uint32_t dataSize)

[Params]

pData [in]: Pointer to BerxelHawkNetParams structure containing network parameters

dataSize [in]: Size of the structure

[Return value]

0: Success

Non-zero: Failure

[Remarks]

BerxelHawkNetParams is defined in BerxelHawkDefines.h.

3.2.24. reconnect Net Device

[Description]

Reconnects a network camera after a TCP disconnection.

[Function]

int32_t reconnectNetDevice()

[Return value]

0: Success

Non-zero: Failure

3.2.25. set Denoise Status

[Description]

Enables or disables the denoising feature. Denoising is enabled by default.

[Function]

int32_t setDenoiseStatus(bool bEnable)

[Params]

bEnable [in]:

true to enable denoising;

false to disable denoising

[Return value]

0: Success

Non-zero: Failure

3.2.26. set Temperature Compensation Enable

[Description]

Enables or disables temperature compensation. This feature is disabled by default.

[Function]

int32_t setTemperatureCompensationEnable(bool bEnable)

[Params]

bEnable [in]:
true to enable compensation;
false to disable compensation

[Return value]

0: Success

Non-zero: Failure

3.2.27. set Frame Sync

[Description]

Enables or disables frame synchronization. Frame sync is enabled by default.

[Function]

int32_t setFrameSync(bool bEnable)

[Params]

bEnable [in]:
true to enable frame sync;
false to disable frame sync

[Return value]

0: Success

Non-zero: Failure

3.2.28. enablen Color Auto Exposure

[Description]

Enables auto-exposure (AE) for color stream.

[Function]

int32_t enableColorAutoExposure()

[Params]

NULL

[Return value]

0: Success

Non-zero: Failure

3.2.29. set Color Exposure Gain

[Description]

Sets the exposure time and gain for the color image.

[Function]

int32_t setColorExposureGain(uint32_t exposureTime, uint32_t gain)

[Params]

exposureTime [in]: Exposure time for color image

Allowed range:

- iHawk137 devices: [1000, 2000, 3000, ..., 10000]
- Other devices: [1000–20000]

gain [in]: Gain value for color image.

See section 3.4.8 BerxelHawkColorGainTable for details.

[Return value]

0: Success

Non-zero: Failure

3.2.30. get Color Exposure Gain

[Description]

Retrieves the current exposure time, gain, and AE status of the color stream.

[Function]

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

[Params]

exposureTime [out]: Current exposure time

gain [out]: Current gain value

aeStatus [out]: AE status (1: enabled, 0: disabled)

[Return value]

0: Success

Non-zero: Failure

3.2.31. set Depth AE Status

[Description]

Sets auto-exposure(AE) functionality for the depth image.

[Function]

int32_t setDepthAEStatus(bool bEnable)

[Params]

bEnable [in]:

true — enable AE

false — disable AE

[Return value]

0: Success

Non-zero: Failure

3.2.32. get Depth AE Status

[Description]

Retrieves the current auto-exposure (AE) status of the depth stream.

[Function]

int32_t getDepthAEStatus(uint32_t* value)

[Params]

value [out]:

1 — AE enabled

0 — AE disabled

[Return value]

0: Success

Non-zero: Failure

3.2.33. set Depth Gain

[Description]

Sets the gain value for the depth stream.

[Function]

int32_t setDepthGain(uint32_t value)

[Params]

Gain value for depth image; valid range: [1–4]

[Return value]

0: Success

Non-zero: Failure

3.2.34. get Depth Gain

[Description]

Retrieves the current gain value of the depth stream.

[Function]

int32_t getDepthGain(uint32_t* value)

[Params]

value [out]: Pointer to the variable receiving the depth gain value

[Return value]

0: Success

Non-zero: Failure

3.2.35. set Depth Exposure

[Description]

Sets the exposure time for the depth image.

[Function]

int32_t setDepthExposure(uint32_t value)

[Params]

value [in]: Exposure time for the depth stream; valid range: [1–43]

[Return value]

0: Success
Non-zero: Failure

3.2.36. get Depth Exposure

[Description]

Retrieves the current exposure time of the depth image.

[Function]

int32_t getDepthExposure(uint32_t* value)

[Params]

value [out]: Pointer to receive the current exposure time

[Return value]

0: Success

Non-zero: Failure

3.2.37. set Depth Electric Current

[Description]

Sets the current for the depth sensor.

[Function]

int32_t setDepthElectricCurrent(uint32_t value)

[Params]

value [in]: Current value in mA; valid range: [800–2000]

[Return value]

0: Success

Non-zero: Failure

3.2.38. get Depth Electric Current

[Description]

Retrieves the current setting of the depth sensor.

[Function]

int32_t getDepthElectricCurrent(uint32_t* value)

[Params]

value [out]: Pointer to store the current value in mA

[Return value]

0: Success

Non-zero: Failure

3.2.39. set Depth Close Range Default Gain And Exposure

[Description]

Sets default gain and exposure values for close-range depth applications. This function is only supported on 100H series devices.

[Function]

int32_t setDepthCloseRangeDefaultGainAndExposure()

[Params]

NULL

[Return value]

0: Success

Non-zero: Failure

3.2.40. enable Device Slave Mode

[Description]

Sets the device into slave mode. Default mode is master.

[Function]

int32_t enableDeviceSlaveMode(bool bEnable)

[Params]

bEnable [in]:

true — enable slave mode

false — disable slave mode (return to master mode)

[Return value]

0: Success

Non-zero: Failure

3.2.41. get Device Master Slave Mode

[Description]

Retrieves the current master/slave mode of the device.

[Function]

int32_t getDeviceMasterSlaveMode(uint32_t* value)

[Params]

value [out]:

1 — Slave mode

0 — Master mode

[Return value]

0: Success

Non-zero: Failure

3.2.42. set User Gpio Ctrl

[Description]

Controls user-configurable GPIO pins.

[Function]

int32_t setUserGpioCtrl(uint32_t gpio_nr, uint32_t gpio_number)

[Params]

gpio_nr [in]: Register index

gpio_number [in]:

1 — Set high level

0 — Set low level

[Return value]

0: Success

Non-zero: Failure

3.2.43. get User Gpio Ctrl

[Description]

Queries the output level of a user-configurable GPIO pin.

[Function]

int32_t getUserGpioCtrl(uint32_t gpio_nr, uint32_t* gpio_number)

[Params]

gpio_nr [in]: Register index

gpio_number [out]:

1 — High level

0 — Low level

[Return value]

0: Success

Non-zero: Failure

3.2.44. set Stream Status

[Description]

Sets whether to report stream data. Use this API after stream startup to pull frames on demand and reduce CPU load.

[Function]

int32_t setStreamStatus(bool bEnable)

[Params]

bEnable [in]:

true — Enable stream reporting

false — Disable stream reporting

[Return value]

0: Success

Non-zero: Failure

3.2.45. set Bind Cpu Core

[Description]

Binds CPU-intensive operations (denoising, registration, decoding) to specific CPU cores.

[Function]

int32_t setBindCpuCore(uint32_t pData[], uint32_t nSize)

[Params]

pData[in]: Array of core indices to bind, e.g.,{0, 2, 3} binds to core 0, 2, 3

nSize [in]: Size of pData

[Return value]

0: Success

Non-zero: Failure

3.2.46. set Depth Remote Mode

[Description]

Configures whether long-range devices display only close-range depth values. Default: true.

[Function]

int32_t setDepthRemoteMode(bool bEnable)

[Params]

bEnable [in]:

true — Max depth value is 8191 (long-range mode)

false — Max depth value is 4095 (standard range)

[Return value]

0: Success

Non-zero: Failure

[Remarks]

For long-range devices with BerxelHawkPixelType = BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D. Refer to sections 3.4.1 and 4.8 for pixel format details.

3.2.47. set Depth Confidence

[Description]

Sets the depth confidence value.

[Function]

int32_t setDepthConfidence(uint32_t nValue)

[Params]

nValue [in]: Depth confidence level; valid range: [3–5]

[Return value]

0: Success

Non-zero: Failure

3.2.48. get Depth Confidence

[Description]

Retrieves the current depth confidence value.

[Function]

int32_t getDepthConfidence(uint32_t* nValue)

[Params]

nValue [out]: Pointer to receive current depth confidence value

[Return value]

0: Success

Non-zero: Failure

3.2.49. set Edge Optimization Status

[Description]

Enables or disables edge optimization. Default is disabled.

[Function]

int32_t setEdgeOptimizationStatus(bool bEnable)

[Params]

bEnable [in]:
true — Enable edge optimization
false — Disable edge optimization

[Return value]

0: Success

Non-zero: Failure

3.2.50. enable Hight Fps Mode

[Description]

Enables or disables high frame rate mode in firmware. Default is disabled.

[Function]

int32_t enableHightFpsMode(bool bEnable)

[Params]

bEnable [in]:
true — Enable high-FPS mode
false — Disable high-FPS mode

[Return value]

0: Success

Non-zero: Failure

3.2.51. set Depth AE Gain Range

[Description]

Sets the auto-exposure (AE) dynamic gain adjustment range for depth images. Default range is [1–4].

[Function]

int32_t setDepthAEGainRange(uint32_t min, uint32_t max)

[Params]

min [in]: Minimum gain value under AE; valid range: [1–4]

max [in]: Maximum gain value under AE; valid range: [1–4]

[Return value]

0: Success

Non-zero: Failure

3.2.52. get Depth AE Gain Range

[Description]

Retrieves the dynamic gain adjustment range under AE for depth images.

[Function]

int32_t getDepthAEGainRange(uint32_t* min, uint32_t* max)

[Params]

min [out]: Pointer to receive minimum AE gain value

max [out]: Pointer to receive maximum AE gain value

[Return value]

0: Success

Non-zero: Failure

3.2.53. set Depth AE Exposure Range

[Description]

Sets the auto-exposure (AE) dynamic exposure adjustment range for depth images.Default range: [1–43] (unit: 0.1 ms)

[Function]

int32_t setDepthAEExposureRange(uint32_t min, uint32_t mid, uint32_t max)

[Params]

min [in]: Minimum exposure value; valid range: [1–43]

mid [in]: Midpoint exposure value; valid range: [1–43]

max [in]: Maximum exposure value; valid range: [1–43]

[Return value]

0: Success

Non-zero: Failure

3.2.54. get Depth AE Exposure Range

[Description]

Retrieves the auto-exposure (AE) dynamic exposure range for depth images. Default range: [1–43] (unit: 0.1 ms)

[Function]

int32_t getDepthAEExposureRange(uint32_t* min, uint32_t* mid, uint32_t* max)

[Params]

min [out]: Pointer to receive minimum exposure value

mid [out]: Pointer to receive midpoint exposure value

max [out]: Pointer to receive maximum exposure value

[Return value]

0: Success

Non-zero: Failure

3.2.55. set Temperature Params File Path

[Description]

Sets the file path for storing temperature compensation parameters. Default path is the SDK library directory.

[Function]

int32_t setTemperatureParamsFilePath(const char* filePath)

[Params]

filePath [in]: Path to temperature parameter file

[Return value]

0: Success

Non-zero: Failure

[Remarks]

Call this function before setTemperaTureCompensationStatus.

Examples：g_pHawkDevice->setTemperatureParamsFilePath("E:\\Data");

3.2.56. set Device TransferMode

[Description]

Sets the USB transfer mode for the device.

[Function]

int32_t setDeviceTransferMode(BerxelHawkUVCMode mode)

[Params]

mode [in]:

BERXEL_HAWK_DEVICE_ISOC_MODE — Isochronous transfer

BERXEL_HAWK_DEVICE_BULK_MODE — Bulk transfer

[Return value]

0: Success

Non-zero: Failure

[Remarks]

Once configured, the device descriptor will be reinitialized and the device will temporarily disconnect. Changes take permanent effect.

3.2.57. set Temporal Denoise Status

[Description]

Enables or disables temporal denoising (inter-frame).

[Function]

int32_t setTemporalDenoiseStatus(bool bEnable)

[Params]

bEnable [in]:

true — Enable temporal denoising

false — Disable

[Return value]

0: Success

Non-zero: Failure

3.2.58. set Spatial Denoise Status

[Description]

Enables or disables spatial denoising (intra-frame).

[Function]

int32_t setSpatialDenoiseStatus(bool bEnable)

[Params]

bEnable [in]:

true — Enable spatial denoising

false — Disable

[Return value]

0: Success

Non-zero: Failure

3.2.59. set Sync Host Time

[Description]

Sets the device clock to synchronize with the host system time.

[Function]

int32_t setSyncHostTime(BerxelHawkSyncTimeType type)

[Params]

type [in]:

BERXEL_HAWK_SYNC_TIME_REALTIME — Synchronize with wall-clock time

BERXEL_HAWK_SYNC_TIME_MONITOR_RAW — Synchronize with host boot time

[Return value]

0: Success

Non-zero: Failure

3.2.60. device Support HwTemp CompenSation

[Description]

Checks whether the device supports hardware-level temperature compensation.

[Function]

bool deviceSupportHwTempCompenSation()

[Params]

None

[Return value]

0: Supported
Non-zero: Not supported

3.2.61. reboot Device

[Description]
Reboots the currently connected device.

[Function]
int32_t rebootDevice()

[Params]
None

[Return value]
0: Success
Non-zero: Failure

3.2.62. get Device Log Size

[Description]
Retrieves the size of the device's internal log buffer.

[Function]
int32_t getDeviceLogSize(uint32_t* size)

[Params]
size [out]: Pointer to receive the size (in bytes) of the available log data

[Return value]
0: Success
Non-zero: Failure

3.2.63. get Device Log

[Description]
Retrieves the log data from the device.

[Function]
int32_t getDeviceLog(void* pData, uint32_t dataSize)

[Params]
pData [out]: Pointer to buffer to receive the log data
dataSize [in]: Size of the buffer

[Return value]
0: Success
Non-zero: Failure

[Remarks]

Use in conjunction with getDeviceLogSize() to ensure the buffer is sufficient. Device must be properly connected during operation.

3.2.64. open Noise Filter

[Description]

Enables or disables the noise filtering function that removes invalid depth points. Disabled by default.

[Function]

int32_t openNoiseFilter(bool bEnable)

[Params]

bEnable [in]:

true — Enable noise filter

false — Disable noise filter

[Return value]

0: Success

Non-zero: Failure

3.2.65. set Max Depth Value

[Description]

Filters out depth values beyond the specified maximum. Unit: mm.

[Function]

int32_t setMaxDepthValue(uint32_t nValue)

[Params]

nValue [in]: Depth threshold in millimeters

[Return value]

0: Success

Non-zero: Failure

[Remarks]

For example, to retain only depth data within 4 meters, set nValue to 4000.

3.2.66. set Filter Ground Value

[Description]

Filters point cloud data by region. Only retains points where X ∈ [leftX, rightX] and Y < groundY. Unit: mm.

[Function]

int32_t setFilterGroundValue(BerxelHawkFilterRange range)

[Params]

range [in]: Structure containing leftX, rightX, and groundY in mm

[Return value]

0: Success

Non-zero: Failure

[Remarks]

For example, to retain data in X: -2000 to +2000 mm and above 190 mm from ground, set range = {-2000, 2000, 190}.

3.2.67. set Depth Optimization Status

[Description]

Sets the depth precision optimization flags.

[Function]

int32_t setDepthOptimizationStatus (bool bOptimizeX, bool bOptimizeY, bool bOptimizeAccuracy)

[Params]

bOptimizeX [in]: Optimize X-direction angles

bOptimizeY [in]: Optimize Y-direction angles

bOptimizeAccuracy [in]: Enable accuracy enhancement

[Return value]

0: Success

Non-zero: Failure

3.3. BerxelHawkFrame Module Description

This module provides access to frame-level data. Refer to BerxelHawkFrame.h.The following functions are provided:

Function Name	Description
BerxelHawkPixelType getPixelType()	Gets the pixel format of the frame
BerxelHawkStreamType getStreamType()	Gets the stream type of the frame
uint32_t getFrameIndex()	Gets the frame index
uint64_t getTimeStamp()	Gets the timestamp
uint32_t getFPS()	Gets the frame rate
uint32_t getWidth()	Gets the frame width

uint32_t getHeight()	Gets the frame height
void* getData()	Gets the frame data pointer
uint32_t getDataSize()	Gets the size of the frame data

3.4. BerxelHawkDefine Module Description

This module defines enums and data structures required by the SDK API. Refer to BerxelHawkDefines.h.

3.4.1. BerxelHawkPixelType Enumeration

This enum defines pixel formats. Details are as follows:

Description	Meaning
BERXEL_HAWK_PIXEL_TYPE_IMAGE_RGB24	Color image format, 3 bytes per pixel (RGB888)
BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D	Depth image format, 16 bits per pixel; upper 12 bits = integer, lower 4 bits = decimal
BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D	Depth image format, 16 bits per pixel; upper 13 bits = integer, lower 3 bits = decimal
BERXEL_HAWK_PIXEL_TYPE_IR_16BIT	Infrared image format, 2 bytes per pixel
BERXEL_HAWK_PIXEL_INVALID_TYPE	Unsupported pixel format

3.4.2. BerxelHawkStreamType Enumeration

Defines stream types used when starting/stopping streams:

Description	Explanation
BERXEL_HAWK_COLOR_STREAM	Color stream
BERXEL_HAWK_DEPTH_STREAM	Depth stream
BERXEL_HAWK_IR_STREAM	Flood IR stream
BERXEL_HAWK_LIGHT_IR_STREAM	Point IR stream
BERXEL_HAWK_INVALID_STREAM	Unsupported stream type

3.4.3. BerxelHawkStreamFlagMode Enumeration

This enumeration defines stream mode types. In SINGULAR mode, only one stream type can be enabled at a time. In MIX mode, multiple stream types can be enabled simultaneously. Details are as follows:

Description	Explanation
BERXEL_HAWK_SINGULAR_STREAM_FLAG_MODE	Single-stream mode; only one stream type supported at a time
BERXEL_HAWK_MIX_STREAM_FLAG_MODE	Mixed VGA stream mode; supports multiple concurrent stream types
BERXEL_HAWK_MIX_HD_STREAM_FLAG_MODE	Mixed HD stream mode; supports multiple concurrent stream types
BERXEL_HAWK_MIX_QVGA_STREAM_FLAG_MODE	Mixed QVGA stream mode; supports multiple concurrent stream types

Supported resolutions under each stream mode (landscape devices):

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@8fps	640@400@30fps
BERXEL_HAWK_MIX_STREAM_FLAG_MODE	640*400@5fps 640*400@10fps 640*400@15fps 640*400@20fps 640*400@25fps 640*400@30fps 1920*1080@5fps 1920*1080@10fps 1920*1080@15fps 1920*1080@20fps 1920*1080@25fps 1920*1080@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_STREAM_FLAG_MODE	1280*800@8fps	1280*800@8fps	Not supported

BERXEL_HAWK_MIX_QVGA_STREAM_FLAG_MODE	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	320*200@5fps 320*200@10fps 320*200@15fps 320*200@20fps 320*200@25fps 320*200@30fps	Not supported

Supported resolutions under each stream mode (portrait devices):

Stream Mode	Color Resolution	Depth Resolution	IR Resolution
BERXEL_HAWK_SINGULAR_STREAM_FLAG_MODE	1080*1920@30fps 400*640@30fps	200*320@5fps 200*320@10fps 200*320@15fps 200*320@20fps 200*320@25fps 200*320@30fps 400*640@5fps 400*640@10fps 400*640@15fps 400*640@20fps 400*640@25fps	400*640@30fps
		400*640@30fps 800*1280@8fps	
BERXEL_HAWK_MIX_STREAM FLAG_MODE	400*640@30fps	400*640@5fps 400*640@10fps 400*640@15fps 400*640@20fps 400*640@25fps 400*640@30fps	400*640@30fps
BERXEL_HAWK_MIX_HD_STREAM_FLAG_MODE	800*1280@10fps	800*1280@8fps	Not supported

BERXEL_HAWK_MIX_QVGA_STREAM_FLAG_MODE	200*320@30fps	200*320@5fps 200*320@10fps 200*320@15fps 200*320@20fps 200*320@25fps 200*320@30fps	Not supported

3.4.4. BerxelHawkDeviceInfo Structure Description

This structure represents device information.It is required when calling functions such as opening/closing devices.

Field Name	Data Type	Description
vendorId	uint16_t	Vendor ID
productId	uint16_t	Product ID
devBus	uint32_t	USB bus number under Linux
devPort	uint32_t	USB port number under Linux
deviceNum	uint32_t	Device number
deviceType	uint32_t	Device type
serialNumber	char[]	Device serial number (available after the device is opened)
deviceAddress	char[]	Device address, used during device opening

3.4.5. BerxelHawkStreamFrameMode Structure Description

This structure represents a stream frame mode. It is used when setting or retrieving FrameMode.

Field Name	Data Type	Description
pixelType	BerxelHawkPixelType	Frame pixel format (see 3.4.1)
resolutionX	int16_t	Horizontal resolution of the frame
resolutionY	int16_t	Vertical resolution of the frame
framerate	int8_t	Frame rate (FPS)

3.4.6. BerxelHawkCameraIntrinsic Structure Description

This structure represents camera intrinsic parameters. These values are used when converting depth to point cloud.：

Field Name	Data Type	Description

fxParam	float	Focal length in X direction
fyParam	float	Focal length in Y direction
cxParam	float	Optical center X coordinate
cyParam	float	Optical center Y coordinate
k1Param	float	Radial distortion coefficient K1
k2Param	float	Radial distortion coefficient K2
p1Param	float	Tangential distortion coefficient P1
p2Param	float	Tangential distortion coefficient P2
k3Param	float	Radial distortion coefficient K3

3.4.7. BerxelHawkDeviceIntrinsicParams Structure Description

This structure contains the device's intrinsic and extrinsic parameters.
The contents of irIntrinsicParams and liteIrIntrinsicParams are identical. See section 3.4.6 for interpretation of intrinsic parameters.

Field Name	Data Type	Description
colorIntrinsicParams	int8_t[]	Color camera intrinsic parameters (9 floats)
irIntrinsicParams	int8_t[]	Flood IR camera intrinsic parameters (9 floats)
liteIrIntrinsicParams	int8_t[]	Point IR camera intrinsic parameters (9 floats)
rotateIntrinsicParams	int8_t[]	Rotation matrix (3×3, 9 floats)
translationIntrinsicPara ms	int8_t[]	Translation vector (3 floats)

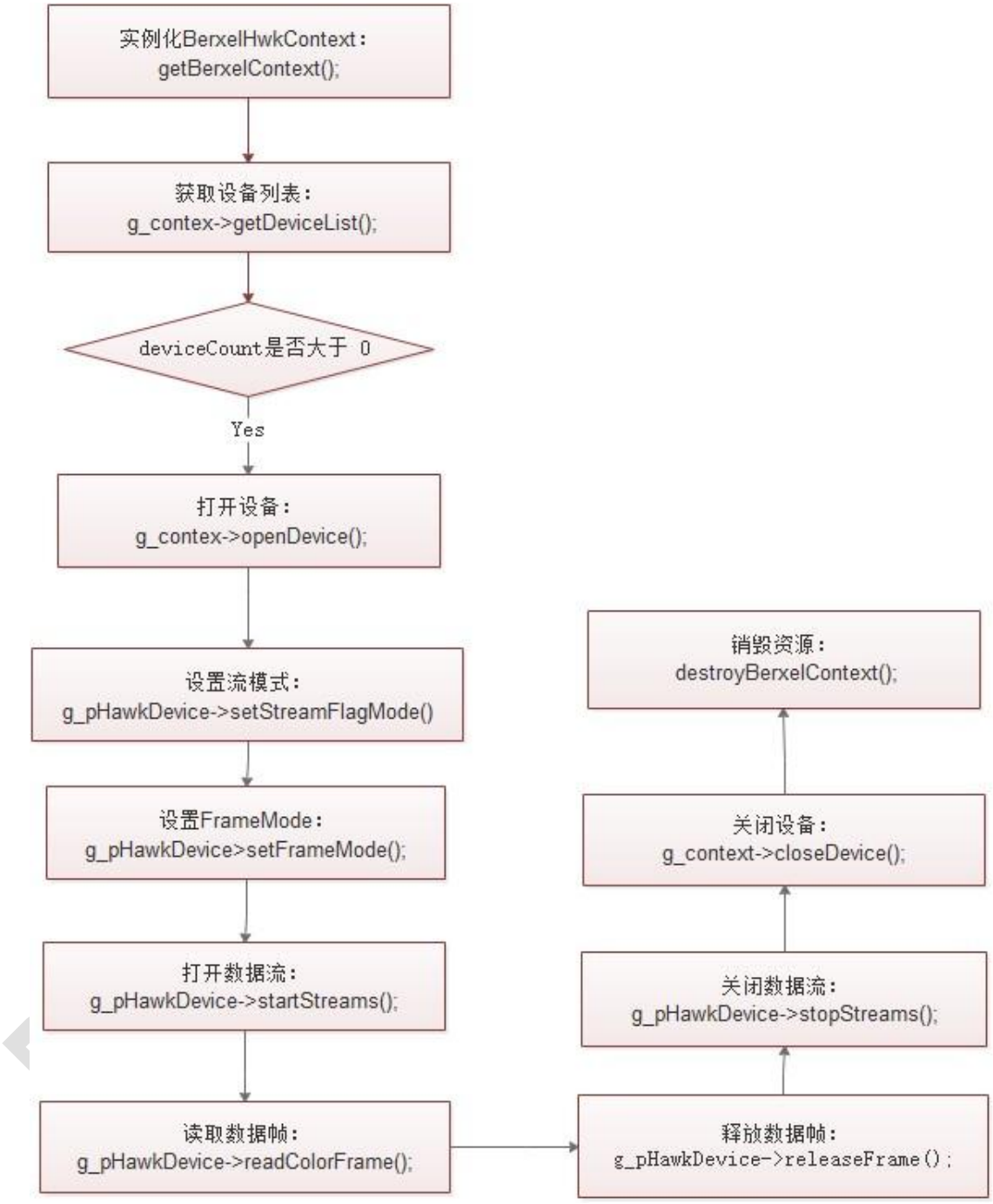
3.4.8. BerxelHawkColorGainTable Description

This table defines gain values required when color AE is disabled. Set the color gain using the values in this array.

```
const uint32_t BerxelHawkColorGainTable[] = {
    100, 101, 103, 104, 106, 107, 109, 110, 112, 114, 115, 117, 118, 120, 121, 123, 125, 126,
    128, 129, 131, 132, 134, 135, 137, 139, 140, 142, 143, 145, 146, 148, 150, 151, 153, 154, 156,
    157, 159, 160, 162, 164, 165, 167, 168, 170, 171, 173, 175, 176, 178, 179, 181, 182, 184,
    185, 187, 189, 190, 192, 193, 195, 196, 198, 200, 203, 206, 209, 212, 215, 218, 221, 225, 228,
    231, 234, 237, 240, 243, 246, 250, 253, 256, 259, 262, 265, 268, 272, 276, 280, 284, 289,
    293, 297, 301, 306, 310, 314, 318, 323, 327, 331, 335, 340, 344, 348, 352, 357, 361, 365, 369,
    374, 378, 382, 386, 391, 395, 399, 403, 408, 412, 416, 420, 425, 429, 433, 437, 442, 446,
    450, 454, 459, 463, 467, 471, 476, 480, 484, 488, 493, 497, 501, 505, 510, 514, 518, 522, 527,
    531, 535, 539, 544, 552, 561, 569, 578, 586, 595, 603, 612, 620, 629, 637, 646, 654, 663,
    671, 680, 688, 697, 705, 714, 722, 731, 739, 748, 756, 765, 773, 782, 790, 799, 807, 816, 824,
    833, 841, 850, 858, 867, 875, 884, 892, 901, 909, 918, 926, 935, 943, 952, 960, 969, 977,
    986, 994, 1003, 1011, 1020, 1028, 1037, 1045, 1054, 1062, 1071, 1079, 1088, 1105, 1122, 1139,
    1156, 1173, 1190, 1207, 1224, 1241, 1258, 1275, 1292, 1309, 1326, 1343, 1360, 1377, 1394,
    1411, 1428, 1445, 1462, 1479, 1496, 1513, 1530, 1547, 1564, 1581, 1598, 1615, 1632, 1649, 1666,
    1683, 1700, 1717, 1734, 1751, 1768, 1785, 1802, 1819, 1836, 1853, 1870, 1887, 1904, 1921,
    1938, 1955, 1972, 1989, 2006, 2023, 2040, 2057, 2074, 2091, 2108, 2125, 2142, 2159, 2176, 2210,
    2244, 2278, 2312, 2346, 2380, 2414, 2448, 2482, 2516, 2550, 2584, 2618, 2652, 2686, 2720,
    2754, 2788, 2822, 2856, 2890, 2924, 2958, 2992, 3026, 3060, 3094, 3128, 3162, 3196, 3230, 3264,
    3298, 3332, 3366, 3400, 3434, 3468, 3502, 3536, 3570, 3604, 3638, 3672, 3706, 3740, 3774,
    3808, 3842, 3876, 3910, 3944, 3978, 4012, 4046, 4080, 4114, 4148, 4182, 4216, 4250, 4284, 4318,
};
```

4. SDK Development Guide

4.1. SDK API Call Flowchart



4.2. Retrieving Color Frames

Refer to the HawkColor sample program under the Samples module of the SDK.

4.3. Retrieving Infrared Frames

Refer to the HawkDepth sample program under the Samples module of the SDK.

4.4. Retrieving Infrared Frames

Refer to the HawkIr sample program under the Samples module of the SDK.

4.5. Retrieving Mixed Color + Depth Frames

Refer to the HawkMixColorDepth sample program under the Samples module of the SDK.

4.6. Retrieving Version Information

The version structure is BerxelHawkVersions, which includes: SDK version; Firmware version and Hardware version. Defined in BerxelHawkDefines.h. Example code:

```
berxel::BerxelHawkVersions   tempVersions;
g_pHawkDevice->getVersion&tempVersions);
```

4.7. Retrieving Current Device Information

The structure BerxelHawkDeviceInfo contains device details including the serial number (SN), vendorId, productId, deviceNum, serialNumber, and deviceAddress. It is defined in the header file BerxelHawkDefines.h. This function should be called after the device is successfully opened. The following is an example of how to retrieve the current device information:

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

4.8. Converting Depth Image Data to Depth Values

There are two formats of depth images. The PixelType obtained via getPixelType from BerxelHawkFrame distinguishes them:

- BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D
- BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D

4.8.1. BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D

Each frame of raw depth data is 16 bits:

- The upper 12 bits represent the integer part (unit: 1 mm)
- The lower 4 bits represent the decimal part (unit: 0.0625 mm)

Conversion steps:

1) // Step 1: Get 16-bit raw depth value
   uint16_t depthOri = pDepth[i];

2) // Step 2: Get integer part
   float depthFront = depthOri >> 4;
3) // Step 3: Get decimal part
   float depthTail = (depthOri & 0x000f) / 16;
4) // Step 4: Combine
   float depth = depthFront + depthTail;

4.8.2. BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D

Each frame of raw depth data is 16 bits:

- The upper 13 bits represent the integer part (unit: 1 mm)
- The lower 3 bits represent the decimal part (unit: 0.125 mm)

Conversion steps:

1) // Step 1: Get 16-bit raw depth value
   uint16_t depthOri = pDepth[i];
2) // Step 2: Get integer part
   float depthFront = depthOri >> 3;
3) // Step 3: Get decimal part
   float depthTail = (depthOri & 0x0007) / 8;
4) // Step 4: Combine
   float depth = depthFront + depthTail;

4.9. Converting Depth Image Data to Point Cloud Data

Converting depth data to point cloud data requires four intrinsic parameters from the camera: focalXParam，focalYParam，opticXParam， opticYParam. Refer to section 3.4.6 for the camera intrinsics definition. Refer to section 3.2.16 for usage of the getCameraIntriscParams function. The typical process is shown below:

```
//Step1： Retrieve depth image
            berxel::BerxelHawkFrame *pHawkFrame = NULL; g_pHawkDevice-
    >readDepthFrame(pHawkFrame,30);

 //Step2：Call convertDepthToPointCloud to convert depth to point cloud
    static berxel::BerxelHawkPoint3D m_3dPoints[400 *640];        g_pHawkDevice-
>convertDepthToPointCloud(pHawkFrame ,1000.0, m_3dPoints);

       For details of convertDepthToPointCloud, refer to section 3.2.13.
```

4.10. Setting Device Status Listener

The SDK currently provides a device status callback interface to monitor device disconnection and connection events. Example code is as follows:

```
//Device status callback function
 void _stdcall Test::onDeviceStatusChange(const char* deviceAddr,const char*
deviceSerialNumber, berxel::BerxelHawkDeviceStatus deviceState, void* pUserData)
      {
                  if(NULL != pUserData)
           {
                Test* pTest = (Test*)pUserData;
      if(NULL != pTest)
                {
     pTest->processDeviceStatusChange(deviceAddr, deviceSerialNumber , deviceState);
                }
           }
      }
      int32_t Test::processDeviceStatusChange(const char* deviceAddr, const char*
 deviceSerialNumber, berxel::BerxelHawkDeviceStatus deviceState){

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

     //Set device status listener
     int32_t Test::init ()
     {
          m_context-
 >setDeviceStateCallback(onDeviceStatusChange,this); }
```

4.11. Retrieving Intrinsic Parameters

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

    berxel::BerxelHawkCameraIntrinsic colorParams;
    berxel::BerxelHawkCameraIntrinsic depthParams;
    float rotateParams[9] = { 0x00 };
    float transParams[3] = { 0x00 };

    memset(&colorParams, 0x00, sizeof(berxel::BerxelHawkCameraIntrinsic));
    memset(&depthParams, 0x00, sizeof(berxel::BerxelHawkCameraIntrinsic));

    // Color camera intrinsics
    memcpy((uint8_t*)&colorParams,intrinsicParams.colorIntrinsicParams,sizeof(berxel::BerxelHawkCameraIntrinsic));

    // Depth camera intrinsics
    memcpy((uint8_t*)&depthParams, intrinsicParams.liteIrIntrinsicParams,
sizeof(berxel::BerxelHawkCameraIntrinsic));

    // Rotation matrix
    memcpy(rotateParams,intrinsicParams.rotateIntrinsicParams, sizeof(rotateParams));

    // Translation matrix
    memcpy(transParams,intrinsicParams.translationIntrinsicParams,sizeof(transParams));
```

4.12. SDK Common Error Codes

Error Code	Description
0	Operation successful
-1	Operation failed
-2	Not initialized
-3	Device not opened

-4	Invalid parameter
-5	Invalid operation
-6	Standard protocol send error
-7	Private protocol send error
-8	Unsupported property operation
-9	Device channel error
-10	Device disconnected
-11	Frame read timeout
-12	Driver not supported
-13	Stream failed to open

5. C#SDK Description

5.1. C#SDK Overview

The C# SDK is located in the directory: Berxel\\BerxelSDK\\C#. It mainly includes the following two parts:

lib	Compiled C# SDK driver
Samples	C# demo program, can be compiled directly using VS2010

5.2. C# SDK Development Instructions

Reference Berxel\\BerxelSDK\\C#\\lib\\BerxelSdkDriver.dll. Since BerxelSdkDriver.dll depends on libraries under Berxel\\BerxelSDK\\lib\\, you must copy all .lib files from that folder to the output directory of your compiled executable.

6. Ros SDK Description

6.1. Ros SDK Overview

The ROS SDK source code is located in: Berxel\\BerxelSDK\\RosSDKNode\\berxel_camera

Node Name	Topic Name	Notes
berxel_camera	/berxel_camera/depth/depth_raw	Depth data
	/berxel_camera/rgb/color_raw	Color data
	/berxel_camera/ir/ir_raw	IR data
	/berxel_camera/berxel_cloudpoint	Point cloud

6.2. ROS SDK Development Instructions

1) Create a ROS workspace. Execute the following command: mkdir -p ~/catkin_work/src Then copy the SDK package into the ~/catkin_work/src directory and extract it.

2) Build the ROS node: cd ~/catkin_work && catkin_make install
3) After building, set the ROS environment variables: source ~/catkin_work/install/setup.bash。
4) Configure the launch parameters in the node parameter file.
For example, to publish the depth image

```
<launch>
    <arg name="serial_no"                   default="" />
    <arg name="usb_bus"                     default="0" />
    <arg name="usb_port"                    default="" />
    <arg name="camera"                      default="berxel_camera" />
    <arg name="tf_prefix"                   default="$(arg camera)"/>
    <arg name="stream_flag"                 default="1" />
    <arg name="stream_type"                 default="2" />
    <arg name="color_width"                 default="640" />
    <arg name="color_height"                default="400" />
    <arg name="depth_width"                 default="640" />
    <arg name="depth_height"                default="400" />
    <arg name="ir_width"                    default="640" />
    <arg name="ir_height"                   default="400" />
    <arg name="depth_fps"                   default="30" />
    <arg name="enable_align"                default="false" />
    <arg name="enable_pointcloud"           default="true" />
    <arg name="enable_color_pointcloud"     default="false" />
    <arg name="enable_denoise"              default="false" />
    <arg name="enable_temperature_compensation"  default="false" />
    <arg name="enable_distance_check"       default="true" />
    <arg name="enable_ordered_pointcloud"   default="true" />
    <arg name="enable_device_timestamp"     default="true" />

    <include file="$(find berxel_camera)/launch/include/berxel_camera_iHawk100Q.launch.xml">
        <arg name="serial_no"                   value="$(arg serial_no)" />
        <arg name="usb_bus"                     value="$(arg usb_bus)"/>
        <arg name="usb_port"                    value="$(arg usb_port)" />
        <arg name="camera"                      value="$(arg camera)" />
        <arg name="tf_prefix"                   value="$(arg tf_prefix)"/>
        <arg name="stream_flag"                 value="$(arg stream_flag)" />
        <arg name="stream_type"                 value="$(arg stream_type)" />
        <arg name="color_width"                 value="$(arg color_width)" />
        <arg name="color_height"                value="$(arg color_height)" />
        <arg name="depth_width"                 value="$(arg depth_width)" />
        <arg name="depth_height"                value="$(arg depth_height)" />
        <arg name="depth_fps"                   value="$(arg depth_fps)" />
        <arg name="enable_align"                value="$(arg enable_align)" />
        <arg name="enable_pointcloud"           value="$(arg enable_pointcloud)" />
        <arg name="enable_color_pointcloud"     value="$(arg enable_color_pointcloud)" />
        <arg name="enable_denoise"              value="$(arg enable_denoise)" />
        <arg name="enable_temperature_compensation"     value="$(arg enable_temperature_compensation)" />
        <arg name="enable_distance_check"       value="$(arg enable_distance_check)" />
        <arg name="enable_ordered_pointcloud"   value="$(arg enable_ordered_pointcloud)" />
        <arg name="enable_device_timestamp"     value="$(arg enable_device_timestamp)" />
    </include>
</launch>
```

topic:

5) Run the compiled berxel_camera node： roslaunch berxel_camera berxel_camera.launch
6) To view data published by the berxel_camera node, use rqt_image_view or rviz. Run: rqt_image_view. Then select the depth topic.
7) To check the published Camera_info message, use: rostopic echo [topic name]

8) For parameter details, please refer to the README.md file inside the ROS package.

```
### 参数说明
"serial_no"                    开启指定序列号设备，不填则默认开启设备列表中第一个设备
"usb_bus"                      usb bus number  (int)
"usb_port"                     usb port number (str,  eg: "2" , "1-2", "1-2-3")
"camera"                       指定节点名
"tf_prefix"                    暂未使用，可通过camera 修改frame_id和tf命名
"stream_flag"                  设备开启模式 berxel::BERXEL_HAWK_SINGULAR_STREAM_FLAG_MODE :  开启单个数据流
                                        berxel::BERXEL_HAWK_MIX_STREAM_FLAG_MODE : Mix VGA 模式(即 彩色+深度 640*400分辨率)
                                        berxel::BERXEL_HAWK_MIX_HD_STREAM_FLAG_MODE :  Mix HD 模式(即彩色+深度 1280*800分辨率)
                                        berxel::BERXEL_HAWK_MIX_QVGA_STREAM_FLAG_MODE :  Mix QVGA 模式(即彩色+深度 320*200分辨率)
"stream_type"                  开启的流类型 1 :  彩色
                                        2 :  深度
                                        3 :  深度+彩色
                                        4 :  泛光源红外
                                        5 :  点光源红外
"color_width"                  彩色图像宽度
"color_height"                 彩色图像高度
"depth_width"                  深度图像宽度
"depth_height"                 深度图像高度
"ir_width"                     红外图像宽度
"ir_height"                    红外图像高度
                                泛光源 :  640 * 400
                                点光源 :  1280 * 800
"depth_fps"                    深度帧率
"enable_align"                 深度对齐彩色
"enable_pointcloud"            点云使能开关  true : 开启点云发布  false :  关闭点云发布
"enable_color_pointcloud"      点云贴图彩色使能开关    true : 开启点云贴图彩色功能  false : 关闭点云贴图彩色功能
"enable_denoise"               降噪使能开关    true : 打开降噪开关  false : 关闭降噪开关
"enable_temperature_compensation"  温度补偿使能开关    true : 打开温度补偿功能  false : 关闭温度补偿功能
"enable_distance_check"        距离感应检查使能开关    true : 打开距离感应检测功能   false : 关闭距离感应使能开关
"enable_ordered_pointcloud"    有序点云使能    true : 有序点云  false : 无序点云
"enable_device_timestamp"      时间戳类型    true : 使用图像自带的时间戳    false : 使用ros::Time
```