

Berxel SDK 开发文档

修订历史

版本号	修订章节	修订记录	修订日期	作者
V1.0		初稿	2021.07.28	Allen
V1.1		1 增加获取相机内参接口说明 2 增加设置图像镜像接口说明 2 增加获取点云数据的说明 3 增加深度帧转换深度值的说明	2021.08.02	Allen
V1.2		1 修改深度转点云接口	2021.08.27	Allen
V1.3		1 增加设置配准开关接口 2 增加设置流模式开关接口	2021.09.02	Allen
V1.4		1 增加设置系统时钟接口	2021.09.14	Allen
V2.0.16		1 增加 C#SDK 相关说明	2021.10.18	Allen
V2.0.18		1 修改 closeDevice 接口参数 2 修改 startUpgrade 接口参数	2021.10.19	Allen
V2.0.21		1 修改 3.4.1 枚举说明 2 增加 4.10 设备监听代码说明 3 增加 RosSDK 相关说明	2021.11.02	Allen
V2.0.22		1 增加设置深度图像曝光接口 2 增加设置深度图像增益接口 3 增加 AE 使能接口	2022.05.12	Sun
V2.0.23		1 增加获取深度图像曝光/增益/电流/AE 状态接口 2 增加获取彩色图像曝光/增益/AE 状态接口 3 增加 GPIO 设置和获取接口	2022.07.12	Sun
V2.0.24		1 增加 setStreamStatus 接口 2 增加 setBindCpuCore 接口	2022.11.24	Sun
V2.0.25		1 增加 setDepthRemoteMode 接口	2023.02.22	Allen
V2.0.26		1 增加 AE 状态下曝光时间、增益范围上下限调整接口	2023.03.22	Sun

目录

1.	概述	8
1.1.	SDK 介绍	8
1.2.	SDK 兼容性	8
2.	开发工具包说明	9
2.1.	SDK 开发工具包模块说明	9
2.2.	SDK Sample 说明	9
2.2.1.	例子程序说明	9
2.2.2.	例子程序 UI 显示说明	9
2.3.	SDK 调用说明	10
2.3.1.	Windows SDK API 调用步骤	10
2.3.2.	Linux SDK API 调用步骤	10
3.	SDK 接口说明	10
3.1.	BeroxelHawkContext 模块说明	10
3.1.1.	getBeroxelContext	10
3.1.2.	destroyBeroxelContext	11
3.1.3.	getDeviceList	11
3.1.4.	getDeviceList	11
3.1.5.	openDevice	12
3.1.6.	openDevice	12
3.1.7.	closeDevice	13

3.1.8.	setDeviceStateCallback	13
3.2.	BerxelHawkDevice 模块说明	14
3.2.1.	startStreams	14
3.2.2.	startStreams	14
3.2.3.	stopStreams.....	15
3.2.4.	getSupportFrameModes	15
3.2.5.	setFrameMode	16
3.2.6.	getCurrentFrameMode	16
3.2.7.	readColorFrame	17
3.2.8.	readDepthFrame	17
3.2.9.	readIrfFrame.....	18
3.2.10.	readLightIrfFrame.....	18
3.2.11.	releaseFrame.....	19
3.2.12.	startUpgrade	19
3.2.13.	convertDepthToPointCloud	20
3.2.14.	getVersion.....	20
3.2.15.	getCurrentDeviceInfo	21
3.2.16.	getCameraIntriscParams	21
3.2.17.	getDeviceIntriscParams.....	22
3.2.18.	getNetParams.....	22
3.2.19.	setStreamMirror.....	23
3.2.20.	setRegistrationEnable.....	23

3.2.21.	setStreamFlagMode	23
3.2.22.	setSystemClock.....	24
3.2.23.	setNetParams	24
3.2.24.	reconnectNetDevice.....	25
3.2.25.	setDenoiseStatus.....	25
3.2.26.	setTemperaTureCompensationEnable	25
3.2.27.	setFrameSync	26
3.2.28.	enableColorAutoExposure	26
3.2.29.	setColorExposureGain	26
3.2.30.	getColorExposureGain.....	27
3.2.31.	setDepthAESTatus.....	27
3.2.32.	getDepthAESTatus	28
3.2.33.	setDepthGain	28
3.2.34.	getDepthGain	29
3.2.35.	setDepthExposure	29
3.2.36.	getDepthExposure.....	29
3.2.37.	setDepthElectricCurrent.....	30
3.2.38.	getDepthElectricCurrent	30
3.2.39.	setDepthCloseRangeDefaultGainAndExposure.....	31
3.2.40.	enableDeviceSlaveMode	31
3.2.41.	getDeviceMasterSlaveMode	31
3.2.42.	setUserGpioCtrl	32

3.2.43.	getUserGpioCtrl.....	32
3.2.44.	setStreamStatus	33
3.2.45.	setBindCpuCore.....	33
3.2.46.	setDepthRemoteMode.....	33
3.2.47.	setDepthConfidence.....	34
3.2.48.	getDepthConfidence	34
3.2.49.	setEdgeOptimizationStatus	34
3.2.50.	enableHightFpsMode	35
3.2.51.	setDepthAEGainRange	35
3.2.52.	getDepthAEGainRange	35
3.2.53.	setDepthAEExposureRange.....	36
3.2.54.	getDepthAEExposureRange.....	36
3.2.55.	setTemperatureParamsFilePath	36
3.2.56.	setDeviceTransferMode	37
3.2.57.	setTemporalDenoiseStatus.....	37
3.2.58.	setSpatialDenoiseStatus.....	37
3.2.59.	setSyncHostTime	38
3.2.60.	deviceSupportHwTempCompenSation.....	38
3.2.61.	rebootDevice	38
3.2.62.	getDeviceLogSize	39
3.2.63.	getDeviceLog	39
3.2.64.	openNoiseFilter	39

3.2.65.	setMaxDepthValue	40
3.2.66.	setFilterGroundValue	40
3.2.67.	setDepthOptimizationStatus	40
3.3.	BeroxelHawkFrame 模块说明	41
3.4.	BeroxelHawkDefine 模块说明	41
3.4.1.	BeroxelHawkPixelType 枚举说明	41
3.4.2.	BeroxelHawkStreamType 枚举说明	42
3.4.3.	BeroxelHawkStreamFlagMode 枚举说明	42
3.4.4.	BeroxelHawkDeviceInfo 结构体说明	44
3.4.5.	BeroxelHawkDeviceInfo 结构体说明	45
3.4.6.	BeroxelHawkStreamFrameMode 结构体说明	45
3.4.7.	BeroxelHawkCameraIntrinsic 结构体说明	45
3.4.8.	BeroxelHawkDeviceIntrinsicParams 结构体说明	45
3.4.9.	BeroxelHawkColorGainTable 说明	46
4.	SDK 开发指引	47
4.1.	SDK API 调用流程图	47
4.2.	获取彩色帧	48
4.3.	获取深度帧	48
4.4.	获取红外帧	48
4.5.	获取彩色+深度混合帧	48
4.6.	获取版本号	48
4.7.	获取当前设备信息	48
4.8.	深度图数据转换为深度值	49
4.8.1.	BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D	49
4.8.2.	BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D	49
4.9.	深度图数据转换为点云数据	49
4.10.	设置设备状态监听	50

4.11.	内参获取.....	51
4.12.	SDK 常见错误码说明.....	51
5.	C#SDK 说明.....	52
5.1.	C#SDK 整体说明	52
5.2.	C#SDK 开发说明	53
6.	Ros SDK 说明	53
6.1.	Ros SDK 整体说明	53
6.2.	Ros SDK 开发说明	53
7.	GMSL SDK 使用说明.....	54
7.1.	Demo 运行	55
	Code 说明	55

1. 概述

1.1.SDK 介绍

Beroxel SDK 是基于Beroxel 3D摄像头的软件开发工具包。产品可广泛适用于工业控制、消费类电子等领域中对三维图像有要求的应用场景,支持 Android/Windows/Linux/ROS 等平台。

1.2.SDK 兼容性

- Windows 7/Win10(x86/x64)
- Ubuntu 14 及以上, ARM x86/x64 平台
- USB 2.0, 电流 2A
- RAM 4G 或以上
- 主频 2.2GHz 或以上

2. 开发工具包说明

2.1. SDK 开发工具包模块说明

模块名称	模块说明
Document	SDK 开发文档说明书
Driver	设备 DFU 模式驱动，主要用于升级
Include	SDK 头文件
Lib	SDK 动态库，调用 SDK API 需包含 BeroxelHawk.lib 静态库
Samples	SDK 例子程序源码，使用 vs2010 编写
Thirdparty	SDK 例子程序显示所需的第三方库
Tools	Beroxel SDK 演示工具
C#	C# SDK 接口以及 Demo

2.2. SDK Sample 说明

2.2.1. 例子程序说明

例子程序名称	例子说明
HawkColor	演示获取彩色图流程
HawkDepth	演示获取深度图流程
HawkIr	演示获取红外图流程
HawkMixColorDepth	演示获取 Mix 流的彩色图+深度图流程（分辨率为 400*640）
HawkMixHDColordDepth	演示获取 Mix 流的彩色图+深度图流程（分辨率为 800*1280）
HawkMixColorDepthMatting	演示配准功能

2.2.2. 例子程序 UI 显示说明

例子程序中使用的是 OpenGL 进行界面展示，在不同平台包含 GL 库及头文件需要注意，作如下说明：

- Windows：需要 freeglut 库，库文件与头文件位于 SDK 安装目录 Thirdparty 下面的 freeglut 文件里面
- Ubuntu：需在当前系统中手动安装 freeglut 库，执行如下指令：
sudo apt-get install freeglut3-dev

2.3.SDK 调用说明

2.3.1.Windows SDK API 调用步骤

- 包含 SDK 安装目录中 lib 文件夹里面的 BernelHawk.lib 静态库
- 包含 SDK 安装目录中 Include 文件夹下面的所有头文件
- 将 lib 库下面的所有文件拷贝到工程编译输出目录中

2.3.2.Linux SDK API 调用步骤

- 在 MakeFile 中链接 SDK 目录中 libs 里面的 BernelHawk.so
- export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:XXX ,其中 xxx 表示 libs 的路径

3. SDK 接口说明

3.1.BernelHawkContext 模块说明

BernelHawkContext 模块主要维护设备列表，负责打开和关闭设备。参见 BernelHawkContext.h 头文件。

3.1.1. getBernelContext

[Description]

获取 BernelHawkContext 对象实例

[Function]

```
static BernelHawkContext* getBernelContext()  
static BernelHawkContext* getBernelContext(const char* strLogPath)
```

[Params]

strLogPath[in] : SDK 日志存放路径

[Return value]

BernelHawkContext 对象实例 : 创建成功
NULL: 创建失败

3.1.2. destroyBernelContext

[Description]

销毁 BernelHawkContext 对象实例

[Function]

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

[Params]

pBernelContext [in] : BernelHawkContext 指针

[Return value]

无

3.1.3. getDeviceList

[Description]

获取设备列表

[Function]

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

[Params]

pDeviceList [out]: 设备列表

pDeviceCount [out]: 设备列表数量

[Return value]

0 : 成功

非 0: 失败

[Remarks]

BernelHawkDeviceInfo 为设备信息结构体，参见 BernelHawkDefines.h 头文件

3.1.4. getDeviceList

[Description]

获取 GMSL 设备列表

[Function]

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

[Params]

pDeviceList [out]: GMSL 设备列表
pDeviceCount [out]: GMSL 设备列表数量

[Return value]

0 : 成功
非 0: 失败

[Remarks]

BernelHawkGmslDeviceInfo 为设备信息结构体, 参见 BernelHawkDefines.h 头文件

3.1.5. openDevice

[Description]

打开设备

[Function]

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

[Params]

deviceInfo[in] : 需打开设备的设备信息结构体, 由 getDeviceList 函数获取

[Return value]

BernelHawkDevice 对象实例 : 打开设备成功
NULL: 打开设备失败

[Remarks]

BernelHawkDevice 为设备管理模块类, 参见 BernelHawkDevice.h 头文件
BernelHawkDeviceInfo 为设备信息结构体, 参见 BernelHawkDefines.h 头文件

3.1.6. openDevice

[Description]

打开 GMSL 设备

[Function]

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

[Params]

deviceInfo[in] : 需打开的 GMSL 设备信息结构体, 由 getDeviceList 函数获取

[Return value]

BernelHawkDevice 对象实例 : 打开设备成功

NULL: 打开设备失败

[Remarks]

BernelHawkDevice 为设备管理模块类, 参见 BernelHawkDevice.h 头文件

BernelHawkGmslDeviceInfo 为设备信息结构体, 参见 BernelHawkDefines.h 头文件

3.1.7. closeDevice

[Description]

关闭设备

[Function]

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

[Params]

hawkDevice[in] : 当前已打开设备的设备句柄

[Return value]

0 : 成功

非 0: 失败

3.1.8. setDeviceStateCallback

[Description]

设置设备状态回调, 以监听设备的断开与连接

[Function]

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

[Params]

Callback[in] : 设备状态回调函数

pUserData[in]: 用户数据, 在回调函数中作为入参传回

[Return value]

0 : 成功

非 0: 失败

[Remarks]

BernelHawkDeviceStatusChangeCallback 为设备状态回调函数，参见 BernelHawkDefines.h 头文件

3.2. BernelHawkDevice 模块说明

BernelHawkDevice 模块主要维护设备相关的操作，包括开/关流、读取数据流、设备信息等，参见 BernelHawkDevice.h 头文件。

3.2.1. startStreams

[Description]

打开数据流

[Function]

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

[Params]

streamFlags[in] : 所需打开流类型的值

[Return value]

0 : 成功
非 0: 失败

[Remarks]

BernelHawkStreamType 为流类型枚举，参见 3.4.2 章节。

3.2.2. startStreams

[Description]

打开数据流，并增加新数据帧到达通知函数

[Function]

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

[Params]

streamFlags[in] : 所需打开流类型的值
callBack[in]: 新数据帧到达通知函数

pUserData[in]: 用户数据指针, BeroxelHawkNewFrameCallBack 回调函数的入参

[Return value]

0 : 成功
非 0: 失败

[Remarks]

BeroxelHawkStreamType 为流类型枚举, 参见 3.4.2 章节。

3.2.3. stopStreams

[Description]

关闭数据流

[Function]

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

[Params]

streamFlags[in] : 所需关闭流类型的值, 使用 BeroxelHawkStreamType 定义的流类型

[Return value]

0 : 成功
非 0: 失败

[Remarks]

BeroxelHawkStreamType 为流类型枚举, 参见 3.4.2 章节。

3.2.4. getSupportFrameModes

[Description]

获取数据流所支持的帧模式列表

[Function]

```
int32_t getSupportFrameModes(BeroxelHawkStreamType streamType,  
    const BeroxelHawkStreamFrameMode** pModes, uint32_t* pCount)
```

[Params]

streamType[in]: 流类型
pModes[out]: FrameMode 列表的指针
pCount [out]: FrameMode 的数量

[Return value]

0 : 成功

非 0: 失败

[Remarks]

BernelHawkStreamType: 参照 3.4.2 章节。

BernelHawkStreamFrameMode: 参照 3.4.5 章节。

3.2.5. setFrameMode

[Description]

设置帧模式

[Function]

```
int32_t setFrameMode(BernelHawkStreamType streamType,  
    BernelHawkStreamFrameMode *mFrameMode)
```

[Params]

streamType[in]: 流类型

mFrameMode[in]: FrameMode指针

[Return value]

0 : 成功

非0: 失败

[Remarks]

BernelHawkStreamType: 参照 3.4.2 章节。

BernelHawkStreamFrameMode: 参照 3.4.5 章节。

3.2.6. getCurrentFrameMode

[Description]

获取当前的帧模式

[Function]

```
int32_t getCurrentFrameMode(BernelHawkStreamType streamType,  
    BernelHawkStreamFrameMode* frameMode)
```

[Params]

streamType[in]: 流类型

frameMode[out]: 当前 FrameMoe 的指针

[Return value]

0 : 成功

非 0: 失败

[Remarks]

BernelHawkStreamType: 参照 3.4.2 章节。

BernelHawkStreamFrameMode: 参照 3.4.5 章节。

3.2.7. readColorFrame

[Description]

读取彩色数据帧

[Function]

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

[Params]

pFrame[out]: 数据帧的类指针

timeout[in]: 超时时间, 若超时, 则pFrame为NULL

[Return value]

0 : 成功

-4: 非法参数

-9: 设备断开

-10: 协议通道异常

-11: 读取数据超时

-13: 非法流句柄 (数据流未打开成功)

-1: 其他错误

[Remarks]

BernelHawkFrame 为数据帧类, 参见BernelHawkFrame.h头文件定义

3.2.8. readDepthFrame

[Description]

读取深度数据帧

[Function]

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

[Params]

pFrame[out]: 数据帧的类指针

timeout[in]: 超时时间, 若超时, 则 pFrame 为 NULL

[Return value]

- 0 : 成功
- 4: 非法参数
- 9: 设备断开
- 10: 协议通道异常
- 11: 读取数据超时
- 13: 非法流句柄（数据流未打开成功）
- 1: 其他错误

[Remarks]

BernelHawkFrame 为数据帧类，参见 BernelHawkFrame.h 头文件定义

3.2.9. readIrFrame

[Description]

读取泛光源红外数据帧

[Function]

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

[Params]

pFrame[out]: 数据帧的类指针

timeout[in]: 超时时间，若超时，则 pFrame 为 NULL

[Return value]

- 0 : 成功
- 4: 非法参数
- 9: 设备断开
- 10: 协议通道异常
- 11: 读取数据超时
- 13: 非法流句柄（数据流未打开成功）
- 1: 其他错误

[Remarks]

BernelHawkFrame 为数据帧类，参见 BernelHawkFrame.h 头文件定义

3.2.10. readLightIrFrame

[Description]

获取点光源红外数据帧

[Function]

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

[Params]

pFrame[out]: 数据帧的类指针

timeout[in]: 超时时间, 若超时, 则 pFrame 为 NULL

[Return value]

0 : 成功

非 0: 失败

[Remarks]

BernelHawkFrame 为数据帧类, 参见 BernelHawkFrame.h 头文件定义

3.2.11. releaseFrame

[Description]

释放读取的数据帧

[Function]

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

[Params]

pHawFrame[in]: 数据帧的指针

[Return value]

0 : 成功

非 0: 失败

[Remarks]

BernelHawkFrame 为数据帧类, 参见 BernelHawkFrame.h 头文件定义

3.2.12. startUpgrade

[Description]

升级固件, 并设置升级状态回调

[Function]

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

[Params]

pCallbacks[in]: 升级状态的回调函数
pUserData[in]: 用户数据, 在回调函数中作为入参传回
pFwFilePath[in] : 设置固件路径

[Return value]

0 : 成功
非0: 失败

[Remarks]

BernelHawkUpgradeProcessCallBack 为升级状态回调函数, 参见
BernelHawkDefines.h头文件定义

3.2.13. convertDepthToPointCloud

[Description]

将深度数据转换为点数据

[Function]

```
int32_t convertDepthToPointCloud(BernelHawkFrame* pFrame , float factor,  
BernelHawkPoint3D* pPointClouds, bool bPointCloudWall = false)
```

[Params]

pFrame [in]: 数据帧
pFactor[in]: 若输出点云坐标以m为单位, 传入1000.0 , 以mm为单位则输入1.0
pPointClouds[out]:输出点云数据
bPointCloudWall[in] : true : 去除点云墙 false : 保留点云墙

[Return value]

0 : 成功
非0: 失败

[Remarks]

BernelHawkPoint3D 为点云数据结构体, 参见 BernelHawkDefines.h 头文件定义

3.2.14. getVersion

[Description]

获取版本号

[Function]

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

[Params]

versions[out]: 版本号信息指针

[Return value]

0 : 成功
非 0: 失败

[Remarks]

BernelHawkVersions 为版本信息结构体, 参见 BernelHawkDefines.h 头文件定义

3.2.15. getCurrentDeviceInfo

[Description]

获取当前打开设备的设备信息

[Function]

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

[Params]

pDeviceInfo[out]: 当前设备信息的指针

[Return value]

0 : 成功
非 0: 失败

[Remarks]

BernelHawkDeviceInfo 为设备信息结构体, 参见 BernelHawkDefines.h 头文件定义

3.2.16. getCameraIntriscParams

[Description]

获取相机红外内参, 此内参可用于点云数据转换

[Function]

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

[Params]

pParams [out]: 相机内参的指针

[Return value]

0 : 成功

非 0: 失败

[Remarks]

BernelHawkCameraIntrinsic 为相机内参结构体，获取的参数是以 400*640（或 640*400）分辨率标定的，参考 3.4.6 章节。

3.2.17. getDeviceIntrinsicParams

[Description]

获取当前设备内外参，包括相机彩色内参，红外内参，平移矩阵，旋转矩阵等参数。

[Function]

```
int32_t getDeviceIntrinsicParams(BernelHawkDeviceIntrinsicParams *pParams )
```

[Params]

pParams [out]: 设备内参的指针

[Return value]

0 : 成功
非 0: 失败

[Remarks]

BernelHawkDeviceIntrinsicParams 为设备内参结构体，相机内参是以 800*1280（或 1280*800）分辨率标注的。参考 3.4.6 章节、3.4.7 章节。

3.2.18. getNetParams

[Description]

获取网络相机详细信息，包括 IP，子网掩码，网关等。

[Function]

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

[Params]

pData: 设备的网络信息(包括 IP, 子网掩码, 网关等, 见结构体 BernelHawkNetParams)
needDataSize: 结构体大小

[Return value]

0 : 成功
非 0: 失败

[Remarks]

BernelHawkNetParams 为设备网络参数结构体，参见 BernelHawkDefines.h 头文件

3.2.19. setStreamMirror

[Description]

设置图像镜像，默认所有数据帧非镜像状态。

[Function]

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

[Params]

bNeedMirror[in]: true 表示为镜像, false 表示为非镜像

[Return value]

0 : 成功
非 0: 失败

3.2.20. setRegistrationEnable

[Description]

设置深度图和彩色图的配准功能是否打开，配准功能默认为关闭状态。

[Function]

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

[Params]

bEnable[in]: true 表示为打开配准, false 表示为关闭配准。

[Return value]

0 : 成功
非 0: 失败

3.2.21. setStreamFlagMode

[Description]

设置流模式，分为单开模式和混合流模式(包含 QVGA、VGA、HD 模式)，默认为混合 VGA 流模式(400*640/640*400 分辨率)。此接口需要在开流接口之前调用。

[Function]

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

[Params]

flagMode[in]: 流模式状态。

[Return value]

0 : 成功
非 0: 失败

3.2.22. setSystemClock

[Description]

同步当前上位机系统时钟到相机中。此函数在开流之前调用, 可以使获取的数据帧的时间戳和上位机的系统时钟同步。

[Function]

int32_t setSystemClock()

[Params]

NULL

[Return value]

0 : 成功
非 0: 失败

3.2.23. setNetParams

[Description]

设置当前设备的 IP 地址, 网关, 子网掩码等参数。

[Function]

int32_t setNetParams(void* pData, uint32_t dataSize)

[Params]

pData[in]: 设备的网络信息(包括 IP, 子网掩码, 网关等, 见结构体 BernelHawkNetParams)

dataSize[in]: 结构体大小

[Return value]

0 : 成功
非 0: 失败

[Remarks]

BernelHawkNetParams 为设备网络参数结构体, 参见 BernelHawkDefines.h

3.2.24. reconnectNetDevice

[Description]

TCP 断开后重新打开设备功能。

[Function]

```
int32_t reconnectNetDevice()
```

[Return value]

0 : 成功
非 0: 失败

3.2.25. setDenoiseStatus

[Description]

设置是否打开当前设备的降噪功能，此功能默认是打开状态。

[Function]

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

[Params]

bEnable[in]: true 表示打开降噪, false 表示关闭降噪, 默认是打开状态。

[Return value]

0 : 成功
非 0: 失败

3.2.26. setTemperatureCompensationEnable

[Description]

设置温度补偿功能是否打开, 温度补偿功能默认为关闭状态。

[Function]

```
int32_t setTemperatureCompensationEnable
```

[Params]

bEnable[in]: true 表示为打开温补功能, false 表示为关闭温补功能。

[Return value]

0 : 成功
非 0: 失败

3.2.27. setFrameSync

[Description]

设置帧同步是否打开，默认打开。

[Function]

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

[Params]

bEnable[in]: true 表示为打开帧同步功能，false 表示为关闭帧同步功能。

[Return value]

0 : 成功
非 0: 失败

3.2.28. enableColorAutoExposure

[Description]

打开彩色 AE 功能

[Function]

```
int32_t enableColorAutoExposure()
```

[Params]

NULL

[Return value]

0 : 成功
非 0: 失败

3.2.29. setColorExposureGain

[Description]

设置彩色图像的曝光时间与增益。

[Function]

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

[Params]

exposureTime[in]: 彩色图像曝光时间, 可设置参数范围:

iHawk137 设备 [1000, 2000, 3000, ..., 10000]

其他设备[1000-20000]

gain[in]: 彩色图像增益, 详情见 3.4.8 节 BernelHawkColorGainTable。

[Return value]

0 : 成功

非 0: 失败

3.2.30. setColorExposureGain

[Description]

获取彩色图像的曝光时间、增益以及 AE 状态。

[Function]

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

[Params]

exposureTime[out]: 彩色图像曝光时间。

gain[out]: 彩色图像增益。

aeStatus: AE 状态 1 打开 0 关闭

[Return value]

0 : 成功

非 0: 失败

3.2.31. setDepthAEStatus

[Description]

设置深度图像 Auto Expoure 功能。

[Function]

```
int32_t setDepthAEStatus(bool bEnable)
```

[Params]

bEnable[in]: true 打开 AE 功能, false 关闭 AE 功能。

[Return value]

0 : 成功
非 0: 失败

3.2.32. getDepthAEStatus

[Description]

获取深度图像 Auto Expoure 状态。

[Function]

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

[Params]

value[out]: 1 AE 功能打开, 0 AE 功能关闭。

[Return value]

0 : 成功
非 0: 失败

3.2.33. setDepthGain

[Description]

设置深度图像的增益。

[Function]

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

[Params]

Value[in]: 深度图像增益值, 可设置参数范围在[1-4]之间。

[Return value]

0 : 成功
非 0: 失败

3.2.34. getDepthGain

[Description]

获取深度图像的增益。

[Function]

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

[Params]

Value[in]: 深度图像增益值。

[Return value]

0 : 成功
非 0: 失败

3.2.35. setDepthExposure

[Description]

设置深度图像的曝光时间。

[Function]

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

[Params]

Value[in]: 深度图像曝光时间，可设置参数范围在[1-43]之间。

[Return value]

0 : 成功
非 0: 失败

3.2.36. getDepthExposure

[Description]

获取深度图像的曝光时间。

[Function]

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

[Params]

Value[out]: 深度图像曝光时间。

[Return value]

0 : 成功
非 0: 失败

3.2.37. setDepthElectricCurrent

[Description]

设置深度 sensor 的电流。

[Function]

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

[Params]

Value[in]: 深度 sensor 的电流(单位 mA), 可设置范围在[800-2000]之间。

[Return value]

0 : 成功
非 0: 失败

3.2.38. getDepthElectricCurrent

[Description]

获取深度 sensor 的电流。

[Function]

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

[Params]

Value[out]: 获取深度 sensor 的电流(单位 mA)。

[Return value]

0 : 成功
非 0: 失败

3.2.39. setDepthCloseRangeDefaultGainAndExposure

[Description]

在近距离使用场景时，设置深度图的默认值 gain 值以及曝光时间，只支持 100H 设备。

[Function]

```
int32_t setDepthCloseRangeDefaultGainAndExposure()
```

[Params]

NULL

[Return value]

0 : 成功
非 0: 失败

3.2.40. enableDeviceSlaveMode

[Description]

设置设备主从模式, 默认主模式。

[Function]

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

[Params]

bEnable[in]: true:设置从模式 false:关闭从模式

[Return value]

0 : 成功
非 0: 失败

3.2.41. getDeviceMasterSlaveMode

[Description]

获取设备主从模式状态。

[Function]

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

[Params]

value[out]: 1:从模式 0:主模式

[Return value]

0 : 成功
非 0: 失败

3.2.42. setUserGpioCtrl

[Description]

设置 Gpio。

[Function]

```
int32_t setUserGpioCtrl(uint32_t gpio_nr, uint32_t gpio_number)
```

[Params]

gpio_nr[in]: 寄存器编号
gpio_number[in]: 1 拉高, 0: 拉低

[Return value]

0 : 成功
非 0: 失败

3.2.43. getUserGpioCtrl

[Description]

设置 Gpio。

[Function]

```
int32_t getUserGpioCtrl(uint32_t gpio_nr, uint32_t* gpio_number)
```

[Params]

gpio_nr[in]: 寄存器编号
gpio_number[out]: 1:高电平, 0:低电平

[Return value]

0 : 成功
非 0: 失败

3.2.44. setStreamStatus

[Description]

设置是否上报数据流 (此接口需求是开流之后通过此接口可以在需要数据的时候去获取数据流, 减少 cpu 压力)。

[Function]

```
int32_t setStreamStatus(bool bEnable)
```

[Params]

true: 开启数据流上报 false: 关闭数据流上报

[Return value]

0 : 成功
非 0: 失败

3.2.45. setBindCpuCore

[Description]

将 SDK 中降噪, 配准, 解码等比较耗 cpu 的算法处理绑定到指定 CPU 核上运行。

[Function]

```
int32_t setBindCpuCore(uint32_t pData[], uint32_t nSize)
```

[Params]

pData[in]: 绑定的 cpu 核编号, 例如 pData[]={0, 2, 3}, 绑定到第一个, 第三个, 第四个核上面运行。

nSize: pData 大小

[Return value]

0 : 成功
非 0: 失败

3.2.46. setDepthRemoteMode

[Description]

设置远距离设备是否只显示近距离深度值, 默认值为 true。

[Function]

```
int32_t setDepthRemoteMode(bool bEnable)
```

[Params]

bEnable[in]: true: 表示远距离设备最大显示深度值为 8191, false: 表示远距离设备最大显示深度值为 4095

[Return value]

0 : 成功
非 0: 失败

[Remarks]

远距离设备的 BernelHawkPixelType 为 BERNEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D, 请参考 3.4.1 章节及 4.8 章节

3.2.47. setDepthConfidence

[Description]

设置深度置信度值。

[Function]

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

[Params]

nValue [in]: 深度置信度, 范围[3-5]

[Return value]

0 : 成功
非 0: 失败

3.2.48. getDepthConfidence

[Description]

获取深度置信度值。

[Function]

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

[Params]

nValue [out]: 获取深度置信度

[Return value]

0 : 成功
非 0: 失败

3.2.49. setEdgeOptimizationStatus

[Description]

设置边缘优化状态, 默认是关闭状态

[Function]

```
int32_t setEdgeOptimizationStatus(bool bEnable)
```

[Params]

bEnable [in]: true : 打开边缘优化 false : 关闭边缘优化

[Return value]

0 : 成功

非 0: 失败

3.2.50. enableHightFpsMode

[Description]

设置固件高帧率模式，默认关闭

[Function]

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

[Params]

bEnable [in]: true : 打开固件高帧率模式 false : 关闭固件高帧率模式

[Return value]

0 : 成功

非 0: 失败

3.2.51. setDepthAEGainRange

[Description]

设置深度图 AE 状态下 Gain 值动态调整范围，设备默认[1-4]

[Function]

```
int32_t setDepthAEGainRange(uint32_t min, uint32_t max)
```

[Params]

min[in]: AE 状态下 Gain 值动态调整范围下限，范围[1-4]

max[in]: AE 状态下 Gain 值动态调整范围上限，范围[1-4]

[Return value]

0 : 成功

非 0: 失败

3.2.52. getDepthAEGainRange

[Description]

获取深度图 AE 状态下 Gain 值动态调整范围

[Function]

```
int32_t getDepthAEGainRange(uint32_t* min, uint32_t* max)
```

[Params]

min[out]: AE 状态下 Gain 值动态调整范围下限

max[out]: AE 状态下 Gain 值动态调整范围上限

[Return value]

0 : 成功

非 0: 失败

3.2.53. setDepthAEEposureRange

[Description]

设置深度图 AE 状态下曝光时间动态调整范围，默认[1~43]，单位 0.1ms

[Function]

```
int32_t setDepthAEEposureRange(uint32_t min, uint32_t mid, uint32_t max)
```

[Params]

min[in]: AE 状态下曝光值动态调整范围下限，范围[1-43]
mid[in]: AE 状态下曝光值动态调整范围中间值，范围[1-43]
max[in]: AE 状态下曝光值动态调整范围上限，范围[1-43]

[Return value]

0 : 成功
非 0: 失败

3.2.54. getDepthAEEposureRange

[Description]

获取深度图 AE 状态下曝光时间动态调整范围，默认[1~43]，单位 0.1ms

[Function]

```
int32_t getDepthAEEposureRange(uint32_t* min, uint32_t* mid, uint32_t* max)
```

[Params]

min[out]: AE 状态下曝光值动态调整范围下限
mid[out]: AE 状态下曝光值动态调整范围中间值
max[out]: AE 状态下曝光值动态调整范围上限

[Return value]

0 : 成功
非 0: 失败

3.2.55. setTemperatureParamsFilePath

[Description]

配置温补参数存放路径，默认路径为 lib 库目录。

[Function]

```
int32_t setTemperatureParamsFilePath(const char* filePath)
```

[Params]

filePath[in]: 所需配置的温补参数路径

[Return value]

0 : 成功

非 0: 失败

[Remarks]

此接口需在调用 setTemperatureCompensationStatus 接口之前调用

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

3.2.56. setDeviceTransferMode

[Description]

设置 USB 设备传输模式

[Function]

```
int32_t setDeviceTransferMode(BernelHawkUVCMode mode)
```

[Params]

mode[in]: 设备传输模式, 同步传输: BERNEL_HAWK_DEVICE_ISOC_MODE
Bulk 传输: BERNEL_HAWK_DEVICE_BULK_MODE

[Return value]

0 : 成功
非 0: 失败

[Remarks]

此接口设置完, 设备会重新配置描述符, 此时设备会断连一下, 设置完成后是永久生效。

3.2.57. setTemporalDenoiseStatus

[Description]

设置帧间降噪状态

[Function]

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

[Params]

bEnable [in]: false : 关闭 true : 打开

[Return value]

0 : 成功
非 0: 失败

3.2.58. setSpatialDenoiseStatus

[Description]

设置空间降噪状态

[Function]

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

[Params]

bEnable [in]: false : 关闭 true : 打开

[Return value]

0 : 成功
非 0: 失败

3.2.59. setSyncHostTime

[Description]

设置同步主机时间

[Function]

int32_t setSyncHostTime(BernelHawkSyncTimeType type)

[Params]

type [in]: BERNEL_HAWK_SYNC_TIME_REALTIME: 主机墙上时间
BERNEL_HAWK_SYNC_TIME_MONITOR_RAW: 主机上电时间

[Return value]

0 : 成功
非 0: 失败

3.2.60. deviceSupportHwTempCompensation

[Description]

设备是否支持硬件温补功能

[Function]

bool deviceSupportHwTempCompensation()

[Params]

无

[Return value]

0 : 成功
非 0: 失败

3.2.61. rebootDevice

[Description]

重启当前设备。

[Function]

int32_t rebootDevice()

[Params]

无

[Return value]

0 : 成功

非 0: 失败

3.2.62. **getDeviceLogSize**

[Description]

获取设备日志大小

[Function]

```
int32_t getDeviceLogSize(uint32_t* size)
```

[Params]

size[out] : 设备日志大小

[Return value]

0 : 成功

非 0: 失败

3.2.63. **getDeviceLog**

[Description]

获取设备日志

[Function]

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

[Params]

pData [out] : 设备日志

dataSize[in]: 日志大小

[Return value]

0 : 成功

非 0: 失败

[Remarks]

此接口配合 getDeviceLogSize 使用，获取设备日志，需保证设备和主句之间通信正常

3.2.64. **openNoiseFilter**

[Description]

设置是否打开去除异常深度值的滤波功能，此功能默认关闭。

[Function]

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

[Params]

bEnable[in]: true 表示打开, false 表示关闭, 默认是关闭状态。

[Return value]

0 : 成功
非 0: 失败

3.2.65. setMaxDepthValue

[Description]

过滤超过某个距离的深度值, 单位是 mm

[Function]

```
int32_t setMaxDepthValue(uint32_t nValue)
```

[Params]

nValue[in]: 过滤的深度值大小

[Return value]

0 : 成功
非 0: 失败

[Remarks]

比如只需要相机 4 米之内的数据, 其他数据不需要, 则将 nValue 设置为 4000

3.2.66. setFilterGroundValue

[Description]

只保留点云中在大于 leftX, 小于 rightX, 同时 y 轴小于 groundY 的数据, 单位是 mm

[Function]

```
int32_t setFilterGroundValue(BernelHawkFilterRange range)
```

[Params]

range[in]: 需过滤数据的结构体, 包含 leftX, rightX 以及 groundY 这 3 个参数

[Return value]

0 : 成功
非 0: 失败

[Remarks]

比如只需要相机在 X 轴-2m 到 2m, 高于地面 19cm 的数据, 则 range 中的 3 个参数可分别设置为-2000, 2000, 190. 这 3 个值

3.2.67. setDepthOptimizationStatus

[Description]

精度优化接口

[Function]

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

[Params]

bOptimizeX [in] : X 角度优化

bOptimizeY [in] : Y 角度优化

bOptimizeAccuracy [in] : 精度优化

[Return value]

0 : 成功

非 0: 失败

3.3. BernelHawkFrame 模块说明

此模块主要是获取读取帧的详细信息，参见 BernelHawkFrame.h 头文件。具体函数说明如下

函数名称	函数说明
BernelHawkPixelType getPixelType()	获取帧图像的象数格式
BernelHawkStreamType getStreamType()	获取帧图像的流类型
uint32_t getFrameIndex()	获取帧图像的帧号
uint64_t getTimeStamp()	获取帧图像时间戳
uint32_t getFPS()	获取帧图像的帧率
uint32_t getWidth()	获取帧图像的 X 轴分辨率
uint32_t getHeight()	获取帧图像的 Y 轴分辨率
void* getData()	获取帧图像数据
uint32_t getDataSize()	获取帧图像的 size 大小

3.4. BernelHawkDefine 模块说明

此模块主要是定义 SDK API 所需的相关枚举以及结构体信息，参见 BernelHawkDefines.h 头文件。

3.4.1. BernelHawkPixelType 枚举说明

此枚举表示像素格式，具体说明如下：

描述	说明
BERXEL_HAWK_PIXEL_TYPE_IMAGE_RGB24	彩色图像像素格式，每个像素占用 3 字节（RGB888）
BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D	深度图像像素格式，每个像素占用 2 字节，

	共 16 位，前 12 位表示整数部分，后 4 位表示小数部分
BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D	深度图像素格式，每个像素占用 2 字节，共 16 位，前 13 位表示整数部分，后 3 位表示小数部分
BERXEL_HAWK_PIXEL_TYPE_IR_16BIT	红外图像素格式，每个像素占用 2 字节
BERXEL_HAWK_PIXEL_INVALID_TYPE	不支持像素格式

3.4.2.BixelHawkStreamType 枚举说明

此枚举表示流类型，在调用开/关流函数中需要传入此参数，具体说明如下：

描述	说明
BERXEL_HAWK_COLOR_STREAM	获取彩色数据的流类型
BERXEL_HAWK_DEPTH_STREAM	获取深度数据的流类型
BERXEL_HAWK_IR_STREAM	获取泛红外数据的流类型
BERXEL_HAWK_LIGHT_IR_STREAM	获取点红外数据的流类型
BERXEL_HAWK_INVALID_STREAM	不支持的流类型

3.4.3.BixelHawkStreamFlagMode 枚举说明

此枚举表示流模式类型，在 SINGULAR 模式下，只支持单一数据流的开关。在 MIX 模式下，支持多种数据流的混合开关。具体说明如下：

描述	说明
BERXEL_HAWK_SINGULAR_STREAM_FLAG_MODE	数据流为单开模式，同时只支持打开一种数据流
BERXEL_HAWK_MIX_STREAM_FLAG_MODE	数据流为混合流 VGA 模式，支持同时打开多种数据流
BERXEL_HAWK_MIX_HD_STREAM_FLAG_MODE	数据流为混合流 HD 模式，支持同时打开多种数据流
BERXEL_HAWK_MIX_QVGA_STREAM_FLAG_MODE	数据流为混合流 QVGA 模式，支持同时打开多种数据流

横版设备各模式下支持分辨率如下：

流模式	彩色分辨率	深度分辨率	红外分辨率
BERXEL_HAWK_SINGULAR_S	1920*1080@30fps	320*200@5fps	640@400@30fps

TREAM_FLAG_MODE	640*400@30fps	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	
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	不支持
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	不支持

竖版设备各模式下支持分辨率如下：

流模式	彩色分辨率	深度分辨率	红外分辨率
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 800*1280@8fps	400*640@30fps
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	不支持
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	不支持

3.4.4. BernelHawkDeviceInfo 结构体说明

此结构体表示设备信息，在开关设备函数中需要传入此结构体信息。具体说明如下：

描述	数据类型	说明
vendorId	uint16_t	供应商识别码
productId	uint16_t	产品识别码
devBus	uint32_t	Linux 下 usb 总线标号
devPort	uint32_t	Linux 下 usb 端口号
deviceNum	uint32_t	设备号
deviceType	uint32_t	设备类型
serialNumber	char[]	设备序列号，在设备打开后才有此信息
deviceAddress	char[]	设备地址，供设备打开时使用

3.4.5. BernelHawkDeviceInfo 结构体说明

此结构体表示设备信息，在开关设备函数中需要传入此结构体信息。具体说明如下：

描述	数据类型	说明
deviceAddress	char[]	设备地址，供设备打开使用
deviceType	char[]	设备类型

3.4.6. BernelHawkStreamFrameMode 结构体说明

此结构体表示帧模式，在设置和获取 FrameMode 中需要使用。具体说明如下：

描述	数据类型	说明
pixelType	BernelHawkPixelType	表示帧像素格式，参照 3.4.1 章节
resolutionX	int16_t	表示帧数据横向分辨率
resolutionY	int16_t	表示帧数据纵向分辨率
framerate	int8_t	表示帧率

3.4.7. BernelHawkCameraIntrinsic 结构体说明

此结构体表示相机内参，相机内参在转换点云数据时候使用，具体说明如下：

描述	数据类型	说明
fxParam	float	X 方向焦距
fyParam	float	Y 方向焦距
cxParam	float	X 方向主光轴点位置
cyParam	float	Y 方向主光轴点位置
k1Param	float	径向畸变 K1
k2Param	float	径向畸变 K2
p1Param	float	切向畸变 p1
p2Param	float	切向畸变 p2
k3Param	float	径向畸变 K3

3.4.8. BernelHawkDeviceIntrinsicParams 结构体说明

此结构体表示设备内参，结构体中 irIntrinsicParams 和 liteIrIntrinsicParams 参数内容一致，相机内参具体说明，参照 3.4.6 章节。

描述	数据类型	说明
colorIntrinsicParams	int8_t[]	彩色相机内参，9 位 float 大小

irIntrinsicParams	int8_t[]	泛红外相机内参，9 位 float 大小
liteIrIntrinsicParams	int8_t[]	点红外相机内参，9 位 float 大小
rotateIntrinsicParams	int8_t[]	旋转矩阵，9 位 float 大小
translationIntrinsicParams	int8_t[]	平移矩阵，3 位 float 大小

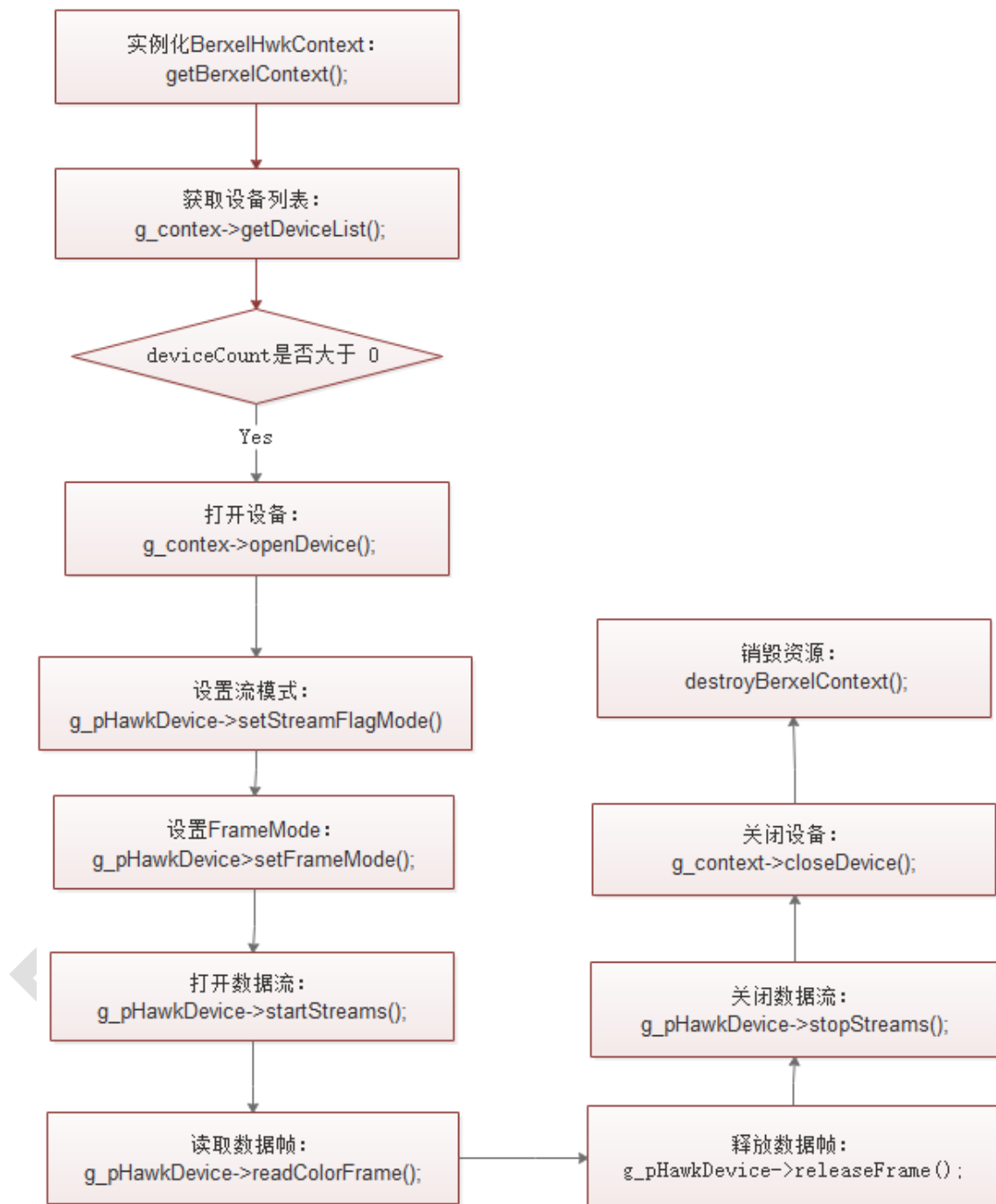
3.4.9. BixelHawkColorGainTable 说明

此表是关闭彩色 AE 功能后需要设置的 gain 值，请按照数组中定义的值设置彩色增益：

```
const uint32_t BixelHawkColorGainTable[] = {
    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 开发指引

4.1. SDK API 调用流程图



4.2. 获取彩色帧

参照 SDK 开发工具包 Samples 模块中 HawkColor 例子程序。

4.3. 获取深度帧

参照 SDK 开发工具包 Samples 模块中 HawkDepth 例子程序。

4.4. 获取红外帧

参照 SDK 开发工具包 Samples 模块中 HawkIr 例子程序。

4.5. 获取彩色+深度混合帧

参照 SDK 开发工具包中 Samples 中 HawkMixColorDepth 例子程序。

4.6. 获取版本号

版本号结构体为 `BernelHawkVersions`，包括 SDK 版本号、固件版本号、硬件版本号。在 `BernelHawkDefines.h` 头文件中定义。示例代码如下：

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

4.7. 获取当前设备信息

设备信息结构体为 `BernelHawkDeviceInfo`，包含 SN、vendorId、productId、deviceNum、serialNumber、deviceAddress 等信息，在 `BernelHawkDefines.h` 头文件中定义。此接口在设备打开成功以后调用，示例代码如下：

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


4.8. 深度图数据转换为深度值

深度图有两种格式，根据 `BeroxelHawkFrame` 中 `getPixelType` 接口获取的 `PixelType` 来区分，分别是 `BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D` 和 `BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D`，两者的精度和有效距离不一致。

4.8.1. BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_12I_4D

深度图每帧 Raw 数据共 16 位,前面 12 位为整数部分，单位是 1mm，后面 4 位为小数部分，单位是 0.0625mm。转换步骤如下：

- ① 获取 16 位深度数据: `uint16_t depthOri = pDepth[i]`; (其中 `pDepth` 代表深度数据指针)
- ② 获取整数部分: `float depthFront = depthOri >> 4;`
- ③ 获取小数点部分: `float depthTail = (depthOri & 0x000f)/16;`
- ④ 完整深度值: `float depth = depthFront + depthTail;`

4.8.2. BERXEL_HAWK_PIXEL_TYPE_DEP_16BIT_13I_3D

深度图每帧 Raw 数据共 16 位,前面 13 位为整数部分，单位是 1mm，后面 3 位为小数部分，单位是 0.125mm。转换步骤如下：

- ① 获取 16 位深度数据: `uint16_t depthOri = pDepth[i]`; (其中 `pDepth` 代表深度数据指针)
- ② 获取整数部分: `float depthFront = depthOri >> 3;`
- ③ 获取小数点部分: `float depthTail = (depthOri & 0x0007)/8;`
- ④ 完整深度值: `float depth = depthFront + depthTail;`

4.9. 深度图数据转换为点云数据

深度数据转换为点云数据，需要相机参数中 `focalXParam`，`focalYParam`，`opticXParam`，`opticYParam` 这 4 个参数，相机内参说明参照 3.4.6 章节，获取相机内参参照 3.2.16 章节关于 `getCameraIntriscParams` 函数的相关说明。具体步骤参照如下代码：

```
//Step1: 获取深度图
berxel::BeroxelHawkFrame *pHawkFrame = NULL;
g_pHawkDevice->readDepthFrame(pHawkFrame, 30);

//Step2: 调用convertDepthToPointCloud 函数将深度图转换成点云图
static berxel::BeroxelHawkPoint3D m_3dPoints[400 * 640];
g_pHawkDevice->convertDepthToPointCloud(pHawkFrame, 1000.0, m_3dPoints);
```

说明： convertDepthToPointCloud 函数相关说明请参照3.2.13章节

4.10. 设置设备状态监听

SDK 目前已提供设备状态监听接口，以监听设备的断开和连接状态，具体示例代码如下：

```
//设备状态回调函数
void _stdcall Test::onDeviceStatusChange(const char* deviceAddr,const char*
deviceSerialNumber, bernel::BernelHawkDeviceStatus 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, bernel::BernelHawkDeviceStatus deviceState){

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

//设置设备状态监听
int32_t Test::init ()
{
    m_context->setDeviceStateCallback(onDeviceStatusChange, this);
}
```

4.11. 内参获取

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

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));

//彩色相机内参
memcpy((uint8_t*)&colorParams, intrinsicParams.colorIntrinsicParams,
sizeof(berxel::BernelHawkCameraIntrinsic));

//深度相机内参
memcpy((uint8_t*)&depthParams, intrinsicParams.liteIrIntrinsicParams,
sizeof(berxel::BernelHawkCameraIntrinsic));

//旋转矩阵
memcpy(rotateParams, intrinsicParams.rotateIntrinsicParams, sizeof(rotateParams));

//平移矩阵
memcpy(transParams, intrinsicParams.translationIntrinsicParams,
sizeof(transParams));
```

4.12. SDK 常见错误码说明

错误码	说明
0	操作成功
-1	操作失败
-2	未初始化

-3	设备未打开
-4	非法参数
-5	非法操作
-6	标准协议发送异常
-7	私有协议发送异常
-8	不支持的属性操作
-9	设备通道异常
-10	设备已断开
-11	读取数据流超时
-12	驱动不支持
-13	流未打开成功

5. C#SDK 说明

5.1. C#SDK 整体说明

C#SDK 位于 Bixel\BixelSDK\C# 目录下面，具体包括如下两部分

lib	封装的 C#SDK Driver
Samples	C#的例子演示程序，使用 VS2010 可直接编译

5.2. C#SDK 开发说明

- ① 首先需引用 Berxel\BeroxelSDK\C# 目录下面的 lib 文件夹里面的 BerxelSdkDriver.dll。
- ② 因 BerxelSdkDriver.dll 依赖 Berxel\BeroxelSDK\lib\ 文件夹里面的库文件，故需将此文件夹下面的所有 lib 库拷贝到工程所生成的 exe 路径里。

6. Ros SDK 说明

6.1. Ros SDK 整体说明

Ros SDK 源码位于 Berxel\BeroxelSDK\RosSDKNode\berxel_camera 目录下:

节点	话题	备注
berxel_camera	/berxel_camera/depth/depth_raw	深度数据
	/berxel_camera/rgb/color_raw	彩色数据
	/berxel_camera/ir/ir_raw	红外数据
	/berxel_camera/berxel_cloudpoint	点云数据

6.2. Ros SDK 开发说明

- ① 创建 ros 工作空间，命令行执行 `mkdir -p ~/catkin_work/src`，将 SDK 包拷贝到 `~/catkin_work/src` 目录下解压。
- ② 开始编译 ros 节点代码，`cd ~/catkin_work && catkin_make install`。
- ③ 编译完成后需要设置下 ros 的环境变量：`source ~/catkin_work/install/setup.bash`。
- ④ 配置节点运行参数文件，比如需要发布深度图像话题：

```

<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>

```

- ⑤ 执行编译完成的 berxel_camera 节点：
roslaunch berxel_camera berxel_camera.launch
- ⑥ 查看 berxel_camera 节点发布数据，可以使用 ros 自带的 rqt_image_view 或者 rviz 查看，命令行直接运行 rqt_image_view，然后选择 depth 话题即可。
- ⑦ 查看发布的 Camera_info 消息，命令行运行 rostopic echo + 话题名称。
- ⑧ 配置文件中参数：

<pre> ### 参数说明 "serial_no" "usb_bus" "usb_port" "camera" "tf_prefix" "stream_flag" "stream_type" "color_width" "color_height" "depth_width" "depth_height" "ir_width" "ir_height" "depth_fps" "enable_align" "enable_pointcloud" "enable_color_pointcloud" "enable_denoise" "enable_temperature_compensation" "enable_distance_check" "enable_ordered_pointcloud" "enable_device_timestamp" </pre>	<pre> 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 number (int) usb port number (str, eg: "2", "1-2", "1-2-3") 指定节点名 暂未使用，可通过camera 修改frame_id和tf命名 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) 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 彩色图像宽度 彩色图像高度 深度图像宽度 深度图像高度 红外图像宽度 红外图像高度 泛光源 : 640 * 400 点光源 : 1280 * 800 深度帧率 深度对齐彩色 点云使能开关 true : 开启点云发布 false : 关闭点云发布 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 降噪使能开关 true : 打开降噪开关 false : 关闭降噪开关 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 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 有序点云使能 true : 有序点云 false : 无序点云 时间戳类型 true : 使用图像自带的时间戳 false : 使用ros::Time </pre>
---	--

详细信息请参考 Ros 包中 README.MD 文件。

7.GMSL SDK 使用说明

由于 GMSL 设备是基于 I2C 通讯，普通用户没有 I2C 读写操作的权限，所以对于 GMSL 设备的所有操作，需要以 root 权限去操作使用。具体操作如下：

7.1.Demo 运行

1. 从商务处获取 SDK 软件版本，并且解压 SDK 包到主控上

```
~/test$ tar -zxvf BeroxelSDK-Linux-2.0.177.tar.gz
```

2. 进入上面解压的 SDK 包中，拷贝 BeroxelSDK 包中的 libs 下面的所有动态库文件到 /usr/lib/路径下

```
berxel@bsp:~/test$ cd BeroxelSDK-Linux-2.0.177/  
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177$  
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177$ cd libs/  
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177/libs$ sudo cp libBeroxel* /usr/lib/  
[sudo] password for berxel:  
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177/libs$
```

3. 进入 sample/src 路径下，带有 GMSL_开头的表示的是 GMSL 的例子程序源码，分别进入对应目录编译生成可执行程序

```
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177$ cd samples/src/  
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177/samples/src$ ls  
3rd      GMSL_HawkColor      HawkAutoSetNetInfo  HawkColorDoubleDevice  HawkDepthDoubleDevice  HawkLightI  
Common   GMSL_HawkMixHDCoDepth HawkColor            HawkDepth              HawkIr                  HawkMixCol  
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177/samples/src$ cd GMSL_HawkColor/  
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177/samples/src/GMSL_HawkColor$ make
```

4. 进入 sample/bin 路径下，可以看到步骤 3 编译生成的可执行程序

```
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177$ cd samples/bin/  
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177/samples/bin$ ls -l  
total 5640  
-rwxrwxr-x 1 berxel berxel 2882432 Jul 25 02:05 GMSL_HawkColor  
-rwxrwxr-x 1 berxel berxel 2886528 Jul 25 01:43 GMSL_HawkMixHDCoDepth
```

5. 通过 root 权限运行步骤三编译的可执行程序

```
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177/samples/bin$ ls -l  
total 5640  
-rwxrwxr-x 1 berxel berxel 2882432 Jul 25 02:05 GMSL_HawkColor  
-rwxrwxr-x 1 berxel berxel 2886528 Jul 25 01:43 GMSL_HawkMixHDCoDepth  
berxel@bsp:~/test/BeroxelSDK-Linux-2.0.177/samples/bin$ sudo ./GMSL_HawkColorDepth  
[Version 2.0.177] [BeroxelCommandDriver] [PUBLIC 20250725_02:10:12:019]##### cur driver Path = /lib/libBeroxelUvcDriver.so  
[Version 2.0.177] [BeroxelCommandDriver] [PUBLIC 20250725_02:10:12:022]##### Load uvc driver succeed!  
[Version 2.0.177] [BeroxelCommandDriver] [PUBLIC 20250725_02:10:12:024]##### cur driver Path = /lib/libBeroxelNetDriver.so  
[Version 2.0.177] [BeroxelCommandDriver] [PUBLIC 20250725_02:10:12:025]##### Load net driver succeed!  
[Version 2.0.177] [BeroxelCommandDriver] [PUBLIC 20250725_02:10:12:033]##### libusb hotplug register callback success  
[Version 2.0.177] [BeroxelCommandDriver] [PUBLIC 20250725_02:10:12:034]#####  
===== Color:1280*800@fps 27 Depth:1280*800@fps 32  
=====  
gmsl device name : gmslDepth:/dev/video8-0  
gmsl device type : PMSensorCamera  
=====  
depth device path : /dev/video8  
rgb device path : /dev/video8  
[Version 2.0.177] [BeroxelHawk] [PUBLIC 20250725_02:10:12:035]#####  
[Version 2.0.177] [BeroxelCommandDriver] [PUBLIC 20250725_02:10:12:036]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:037]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:038]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:039]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:040]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:041]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:042]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:043]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:044]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:045]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:046]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:047]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:048]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:049]#####  
[Version 2.0.177] [BeroxelUvcDriver] [PUBLIC 20250725_02:10:12:050]#####
```

Code 说明

6. 获取 SDK 上下文

```
//获取SDK上下文
```

```
g_context = berxel::BeroxelHawkContext::getBeroxelContext();
```

7. 获取 GMSL 设备句柄

```
berxel::BeroxelHawkGmslDeviceInfo* pGmslDeviceInfo = NULL;
uint32_t gmslDeviceCount = 0;
//获取GMSL设备列表
g_context->getDeviceList(&pGmslDeviceInfo, &gmslDeviceCount);
if((gmslDeviceCount <= 0) || (NULL == pGmslDeviceInfo))
{
    sprintf(g_errMsg,"%s", "Get No Connected BerxelDevice");
    return creatWindow(argc ,argv);
}

for (int i = 0; i < gmslDeviceCount; i++)
{
    printf("#####\n\n\n");
    printf("gmsl device name : %s\n", pGmslDeviceInfo[i].deviceAddress);
    printf("gmsl device type : %s\n", pGmslDeviceInfo[i].deviceType);
}
```

8. 设置开启 Stream Mode

```
//设置开启Mix HD 模式 RGB流+深度流 1280*800
g_pHawkDevice->setStreamFlagMode(berxel::BERXEL_HAWK_MIX_HD_STREAM_FLAG_MODE);
```

9. 设置帧率和分辨率

```
//设置帧率和分辨率
berxel::BeroxelHawkStreamFrameMode colorFrameMode;
g_pHawkDevice->getCurrentFrameMode(berxel::BERXEL_HAWK_COLOR_STREAM , &colorFrameMode);
g_pHawkDevice->setFrameMode(berxel::BERXEL_HAWK_COLOR_STREAM, &colorFrameMode);
berxel::BeroxelHawkStreamFrameMode depthFrameMode;
g_pHawkDevice->getCurrentFrameMode(berxel::BERXEL_HAWK_DEPTH_STREAM , &depthFrameMode);
g_pHawkDevice->setFrameMode(berxel::BERXEL_HAWK_DEPTH_STREAM, &depthFrameMode);
```

10. 开启数据流

```
//开启数据流
int ret = g_pHawkDevice->startStreams(berxel::BERXEL_HAWK_COLOR_STREAM | berxel::BERXEL_HAWK_DEPTH_STREAM);
if(ret != 0)
{
    sprintf(g_errMsg,"%s", "Open Berxel Stream Failed");
    return creatWindow(argc ,argv);
}
```

11. 读取图像数据

```
if(g_pHawkDevice)
{
    g_pHawkDevice->readDepthFrame(pHawkDepthFrame,30);
    if(pHawkDepthFrame == NULL )
    {
        return false;
    }

    g_pHawkDevice->readColorFrame(pHawkColorFrame,30);
    if(pHawkColorFrame == NULL)
    {
        g_pHawkDevice->releaseFrame(pHawkDepthFrame);
        return false;
    }
}
```

12. 关闭数据流


```
//关闭数据流
if(g_pHawkDevice)
{
    g_pHawkDevice->stopStreams(berxel::BERXEL_HAWK_COLOR_STREAM | berxel::BERXEL_HAWK_DEPTH_STREAM);
}
```

13. 关闭设备

```
//关闭设备
if(g_context)
{
    g_context->closeDevice(g_pHawkDevice);
}
```

14. 销毁 SDK 上下文

```
//销毁SDK上下文
if(g_context)
{
    berxel::BeroxelHawkContext::destroyBeroxelContext(g_context);
}
```