昊博影像 HAOBO IMAGING
上海昊博影像科技有限公司

昊博静态平板探测器
HBI_FPD API Programming Reference

15/01/2021

更新记录

版本号	日期	描述	作者
Ver 1.0.0.0	2019.8.20	文档创建	XM.LI
Ver 1.0.1.4	2019.9.20	更新	MH.YANG
Ver 1.0.1.4	2019.9.23	更新流程图，API 说明	XM.LI
Ver 1.0.1.8	2019.11.04	更新接口	MH.YANG
Ver 1.0.1.9	2019.11.13	更新校正失效	MH.YANG XM.LI
Ver 1.0.1.10	2019.11.25	增加 FPD 状态监控	MH.YANG
Ver 1.0.1.11	2019.11.28	优化和增加日志监控	MH.YANG
Ver 1.0.1.12	2019.12.02	支持双网卡或多网卡	MH.YANG
Ver 1.0.1.13	2019.12.03	回调函数增加帧号	MH.YANG
Ver 1.0.1.13	2019.12.04	增加 sdk 初始化参数接口函数(触发模式、固件校正和软件校正)	MH.YANG XM.LI
Ver 1.0.2.1	2019.12.24	自适应 3543 和 4343，gain 和 defect 增加有效校正区域	MH.YANG XM.LI
Ver 1.0.2.2	2020.01.08	校正算法异常崩溃，offset 校正失败以及校正逻辑优化	MH.YANG XM.LI
Ver 1.0.2.3	2020.02.11	双坏线校正处理	MH.YANG XM.LI
Ver 1.0.2.4	2020.02.11	优化生成模板流程以及操作	MH.YANG XM.LI
Ver 1.0.2.5	2020.02.11	修复 defect 算法	MH.YANG XM.LI
Ver 1.0.2.6	2020.02.11	优化校正算法以及处理异常	MH.YANG XM.LI
Ver 1.0.2.7	2020.02.11	优化校正算法以及处理异常，支持 32 位和 64 位	MH.YANG XM.LI
Ver 1.0.2.8	2020.03.30	1、增加 prpare 状态； 2、重置固件参数； 3、修改监控线程释放 bug； 4、增加 **factor** 配置； 5、增加 HBI_QuckInitDllCfg(可替换函数 HBI_WorkStationInitDllCfg)； 6、添加设置\获取增益模式接口。	MH.YANG
Ver 1.0.2.10	2020.04.02	增加 SN 码配置，打印日志输出信息不正确	MH.YANG
Ver 1.0.2.13	2020.04.17	增加增益设置/获取模式接口; 增加采集设置采集时间间隔/	MH.YANG

		获取接口；增加 AED 积分设置/获取接口；增加校正模板生成说明。	
Ver 1.0.3.1	2020.05.13	偶发连接失败以及环境配置说明。	MH.YANG
Ver 1.0.3.2	2020.05.21	连续采集接收自动切换采集模式以及支持巨帧通讯。	MH.YANG
Ver 1.0.3.3	2020.06.02	处理图像校正使能例外异常情况	MH.YANG
Ver 1.0.3.4	2020.06.04	新增接口函数： **HBI_TriggerAndCorrectApplay** **HBI_UpdateTriggerMode** **HBI_UpdateCorrectEnable** **HBI_GetCorrectEnable** 旧接口函数： **HBI_WorkStationInitDllCfg** **HBI_QuckInitDllCfg**	MH.YANG
Ver 1.0.3.8	2020.06.24	新增接口函数： **HBI_SetSoftwareCorrec** **HBI_GetSoftwareCorrect** 旧接口函数： **HBI_SetEnableCalibrate** **HBI_GetEnableCalibrate** 更新矫正算法	MH.YANG
Ver 1.0.3.9	2020.06.29	新增接口函数： **HBI_SetAedThreshold** **HBI_GetAedThreshold** **HBI_SetSaturationValue** **HBI_GetSaturationValue** **HBI_SetClippingValue** **HBI_GetClippingValue**	MH.YANG
Ver 1.0.3.10	2020.07.06	新增接口函数： **HBI_SetPreAcqTm** **HBI_GetPreAcqTm**	MH.YANG
Ver 1.0.3.11	2020.07.09	支持 1613、2530 和 3030 参数优化	MH.YANG
Ver 1.0.3.12	2020.07.30	设置使能参数导致个别参数异常 固件升级 **gain 和 defect 模板上传**	MH.YANG
Ver 1.0.3.13	2020.08.06	增加新接口 **HBI_SetLPFLevel** **HBI_GetLPFLevel**	MH.YANG
Ver 1.0.3.14	2020.08.14	软触发 perpare 加延时保存异常	MH.YANG

Ver 1.0.3.15	2020.08.18	个别日志打印信息不正确	MH.YANG
Ver 1.0.4.0	2020.08.26	Gain 失校正处理	MH.YANG
Ver 1.0.4.1	2020.09.04	制作 Gain 模板异常	MH.YANG
Ver 1.0.4.2	2020.09.11	单帧采集和连续采集显示并保存原图	MH.YANG
Ver 1.0.4.3	2020.10.20	16 位数据转 8 位数据并自动窗宽窗位调节 **HBI_16UCConvertTo8UC**	MH.YANG
Ver 1.0.4.4	2020.10.20	增加一键生成校正模板接口 **HBI_ImmediateGenerateTemplate**	MH.YANG
Ver 1.0.4.5	2020.10.28	**连续采集丢包重传**	MH.YANG
Ver 1.0.4.6	2020.10.28	增加 preview 预览图像功能： **HBI_SetPreviewMode** 支持 4343 100um 优化重传机制	MH.YANG
Ver 1.0.4.7	2020.11.11	单帧采集丢包重传与 ready 信号不同步问题	MH.YANG
Ver 1.0.4.8	2020.11.18	连续采集丢包问题	MH.YANG
Ver 1.0.4.9	2020.11.27	1、preview 模板生成(包括分布生成和一键生成模板)； 2、preview 图像校正； 3、增加 SAEC 模式 per ready time 和 post ready time 设置； 4、增加友通对接接口； **HBI_SAecModeApply** **HBI_SAecAcq** **5、更新接口(判断是否生成 preview 模板参数)** **HBI_ImmediateGenerateTemplate** **HBI_GenerateOffsetTemp** **HBI_GenerateGainTemp** **HBI_GenerateDefectTemp**	MH.YANG
Ver 1.0.4.10	2020.12.23	1、个别对外接口中数据拷贝有异常，已处理；	MH.YANG
Ver 1.0.4.11	2021.01.15	1、默认连续采集下前 n 帧抛弃问题； 2、设置软件校正状态修改；	MH.YANG

目录

1、介绍

1.1 目的

本文档是昊博影像静态平板探测器的 SDK 开发说明书，该说明书详细的说明了 SDK 函数库接口的功能、参数以及调用流程等，主要目的是方便用户快速的进行二次开发，特提供文档支持。参考的对象是使用昊博影像医用平板探测器进行二次开发的技术人员。

1.2 关于 SDK

为了方便用户集成开发，我们提供标准的动态链接库(DLL)接口。客户程序可以通过导入动态链接库， 对动态链接库中封装的接口程序进行调用，以实现对平板探测器的控制和图像采集。

软件系统总体结构， 接口层主要功能包括：图像采集和图像校正，探测器校准和系统管理，见图 1。

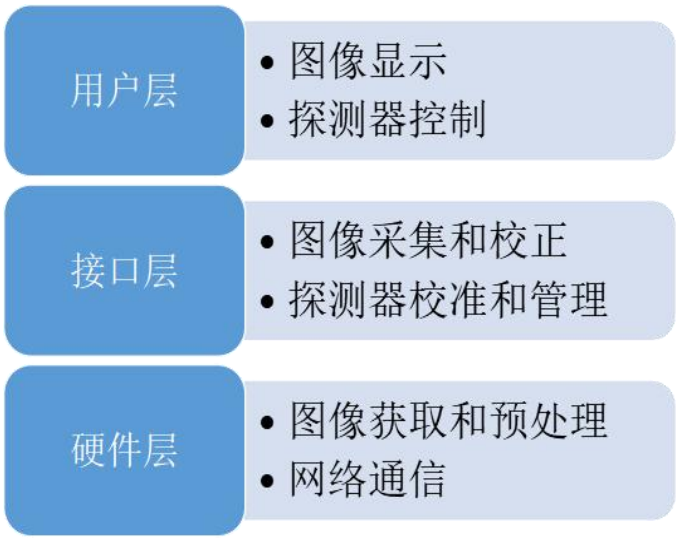


图 1

该接口软件使用 C++ 实现。函数的返回类型为整型，表明函数调用的成功或失败。返回类型为字符串，表明需要返回的是内容提示或数据。
该软件可运行于 32 位或 64 位 windows 操作系统。

1.3 环境

目前支持 32 和 64 位开发。

操作系统： Windows 7 以上，后期会有 Linux 版本推出
内存： 4GB or more
硬盘： >= 10G
网卡： UDP 通讯

1.4 安装

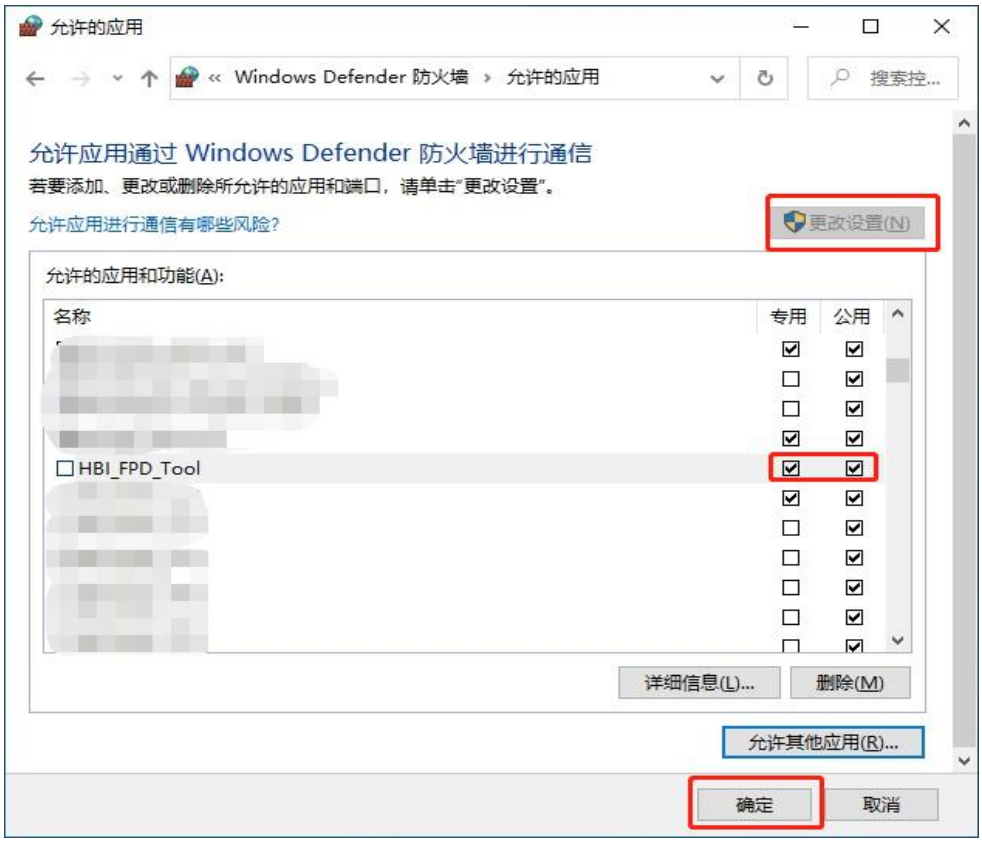
环境配置：

使用软件前配置本地 IP 地址（目前平板的 IP 地址默认为 192.168.10.40），本地网卡默认配置 192.168.10.20，如下图：

注意：第一次打开工具软件，连接时提示禁用或允许时，选择允许，否则可能会出现连接成功失败或者不上图。如果存在这样情况注意检查并配置防火墙，有两种方法：

方法一：直接关闭防火墙；

方法二：防火墙例外设置-“允许应用通过 Windows 防火墙”；



该接口是基于 C/C++开发语言，注意依赖库。

头文件 Including Header Files：

HbFpdDll.h - 接口函数文件

HbDllType.h - 参数、结构体模型文件

HbDllError.h - 错误列表文件

动态库文件 Linking the Library：

生成库：HBI_FPD.lib - 静态库
HBI_FPD.dll - 动态库

依赖库：opencv_core249.dll
opencv_highgui249.dll
opencv_imgproc249.dll

2、概述

2.1 数据流程图

该软件与探测器设备之间的数据可分为两种：命令数据和图像数据。其中命令数据，连接、采集、设置或获取参数等，图像数据亮场图需要与高压发生器联动，数据流程如下，图 2-1 和图 2-2。

命令数据流程图

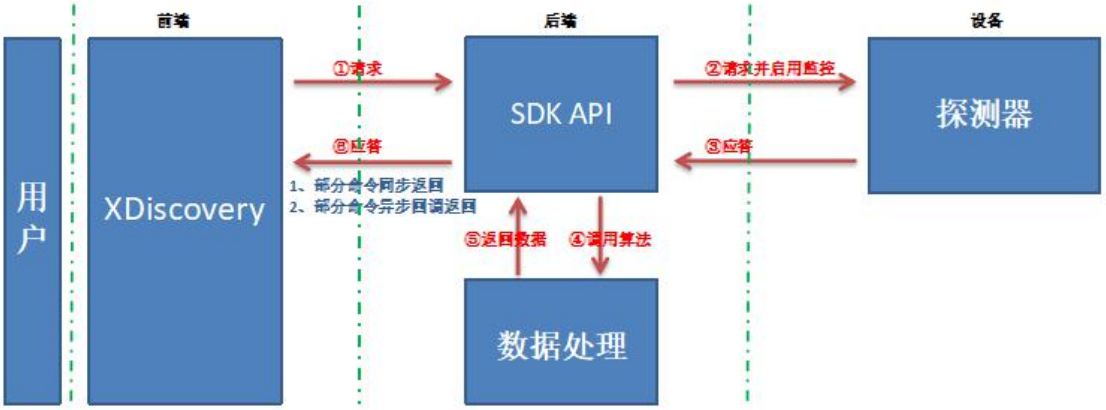


图 2-1

图像数据流程图

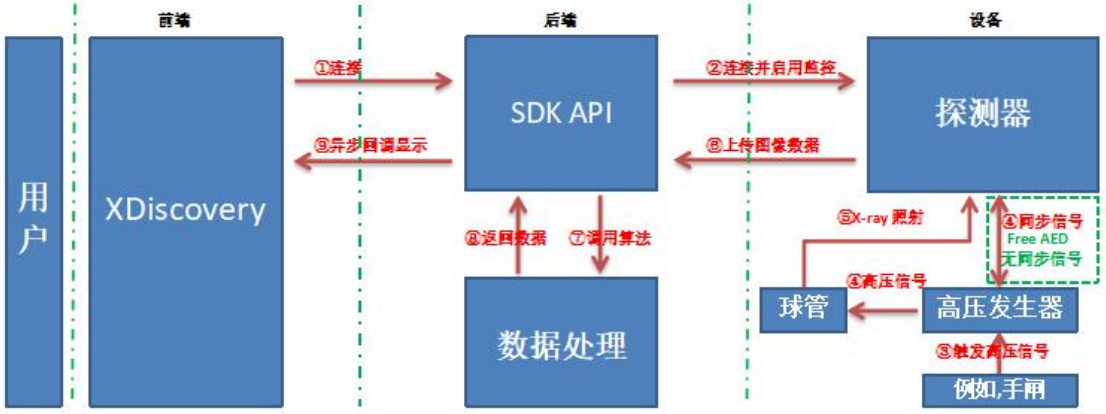
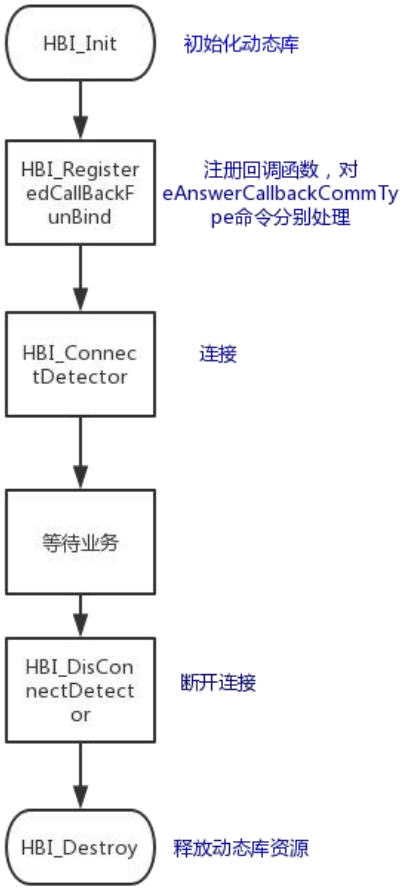


图 2-2

其中，图像数据因为触发模式不同，软触发、高压触发与 Free AED 区别在与是否向探测器同时发出同步信号。

2.2 调用流程图

主流程流程图：



其他操作流程图：

参数设置：

HBI_Set* -> 发送网络命令 -> HBI_Get* -> 获取最新配置

单帧采集：

HBI_SingleAcquisition -> 通知接收线程 -> 接收数据单帧处理

多帧采集：

HBI_LiveAcquisition -> 通知接收线程，传入参数 -> 接收数据多帧处理

多帧采集保存文件：

HBI_SetRawStyle -> HBI_LiveAcquisition -> 通知接收线程 -> 接收数据多帧处理

Offset 模板生成：
配置丢弃帧数 ndiscard 和采集帧 nframesum
->采集图像 HBI_LiveAcquisition(DYNAMIC_ACQ_BARK_MODE)
-> 线程接收数据保存 -> HBI_GenerateDarkingTemp -> 计算保存模板

Gain 模板生成：
生成不同剂量值采集组 -> 配置每组采集帧数 nframesum -> 配置丢弃帧数 ndiscard -> 采集亮场图像(设置采集模式-》高压触发自动采集-》生成校正模型) -> HBI_GenerateBrightTemp->计算生成模板。

Defect 模板生成：
生成不同剂量值采集组 -> 配置每组采集帧数 nframesum -> 配置丢弃帧数 ndiscard -> 采集亮场图像(设置采集模式-》高压触发采集-》生成校正模型) -> HBI_GenerateBrightTemp->计算生成模板。

固件升级暂不提供。

2.3 回调函数

回调函数就是一个通过函数指针调用的函数。如果你把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，我们就说这是回调函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。

回调是任何一个被以方法为其第一个参数的其它方法的调用的方法。很多时候，回调是一个当某些事件发生时被调用的方法。

为了防止 UI 阻塞，部分接口采用了异步接口，使用了注册回调函数，注册函数一般有全局函数和静态成员函数两种：

```
static int handleCommandEvent(unsigned char command, void *buff, int len, int nID);
class CXXX {
    ...
public:
static int handleCommandEvent(unsigned char command, void *buff, int len);
...
};
```

参数说明：
unsigned char command：参考 HbDllType.h 中 enum eCallbackEventCommType；

void *buff：一般为 unsigned char 类型，不同命令对应不同地址；

int len：返回码或缓存长度等。

int nID：图像数据为该图像的帧号。

回调函数返回：

case ECALLBACK_TYPE_NET_ERR_MSG:

case ECALLBACK_TYPE_ROM_UPLOAD:/* 更新配置 */

case ECALLBACK_TYPE_SINGLE_IMAGE: // 单帧采集上图

case ECALLBACK_TYPE_MULTIPLE_IMAGE: // 连续采集上图

例：平板状态

```
enum EFpdStatusType {
	FPD_STATUS_CONN = 0x00,
	FPD_STATUS_DISCONN,
	FPD_STATUS_READY,
	FPD_STATUS_BUSY
};

	case ECALLBACK_TYPE_NET_ERR_MSG: // 平板状态：连接/断开/ready/busy
		printf("ECALLBACK_TYPE_NET_ERR_MSG,recode=%d\n", len);
		if (theDemo != NULL) {
			CString strMsg = _T("");
			if (len <= 0 && len >= -7) {
				if (len == 0)
					printf("ECALLBACK_TYPE_NET_ERR_MSG,Err:网络未连接!\n");
				else if (len == -1)
					printf("ECALLBACK_TYPE_NET_ERR_MSG,Err:参数异常!\n");
				else if (len == -2)
					printf("ECALLBACK_TYPE_NET_ERR_MSG,Err:准备就绪的描述符数返
回失败!\n");
				else if (len == -3)
					printf("ECALLBACK_TYPE_NET_ERR_MSG,Err:接收超时!\n");
				else if (len == -4)
					printf("ECALLBACK_TYPE_NET_ERR_MSG,Err:接收失败!\n");
				else if (len == -5)
					printf("ECALLBACK_TYPE_NET_ERR_MSG,Err:端口不可读!\n");
				else if (len == -6)
					printf("ECALLBACK_TYPE_NET_ERR_MSG,network card unusual!\n");
				else if (len == -7)
					printf("ECALLBACK_TYPE_NET_ERR_MSG,network card ok!\n");
				status = (int)FPD_STATUS_DISCONN;
			}
			else if (len == 1) { // connect
				printf("ECALLBACK_TYPE_NET_ERR_MSG,开始监听!\n");
```

```
            status = (int)FPD_STATUS_CONN;
        }
        else if (len == 2) { // ready
            printf("ECALLBACK_TYPE_NET_ERR_MSG,ready!\n");
            status = (int)FPD_STATUS_READY;
        }
        else if (len == 3) { // busy
            printf("ECALLBACK_TYPE_NET_ERR_MSG,busy!\n");
            status = (int)FPD_STATUS_BUSY;
        }
        else if (len == 4) { // prepare
            printf("ECALLBACK_TYPE_NET_ERR_MSG,prepare!\n");
            status = (int)FPD_STATUS_PREPARE;
        }
        else
            printf("ECALLBACK_TYPE_NET_ERR_MSG,Err:other error=%d\n", len);

        // ADD BY MH.YANG 2019/11/12
        if (status != -1) {
            // 更新图标
            theDemo->PostMessage(WM_USER_CONNECT_STATUS_ICON,
(WPARAM)status, (LPARAM)0);

            // 更新状态信息
            if (len <= 0 && len >= -6) {
                /*
                * 触发断开消息
                */
                if (theDemo->m_pFpd != NULL)
                    HBI_DisConnectDetector(theDemo->m_pFpd);
                // 更新平板断开信息
                theDemo->PostMessage(WM_USER_CURR_FPD_STATUS,
(WPARAM)0, (LPARAM)0);
            }
        }
    }
```

2.4 请求和返回命令

2.4.1 请求命令

```
// Notice:    Request Command
enum eRequestCommType
{
    EDL_COMMON_TYPE_INVALVE                    = 0x00, // 无效命令
```

```
    EDL_COMMON_TYPE_GLOBAL_RESET          = 0x01, //  重置
    EDL_COMMON_TYPE_PREPARE               = 0x02, //
    EDL_COMMON_TYPE_SINGLE_SHORT          = 0x03, //  单帧采集
    EDL_COMMON_TYPE_LIVE_ACQ              = 0x04, //  连续采集
    EDL_COMMON_TYPE_STOP_ACQ              = 0x05, //  停止采集采集
    ... ...
    EDL_COMMON_TYPE_DUMMPING              = 0x08, //  清空
   EDL_COMMON_TYPE_END_RESPONSE                = 0x0A, // End response packet
    EDL_COMMON_TYPE_SET_ROM_PARAM_CFG     = 0x12, //  设置固件参数
    EDL_COMMON_TYPE_GET_ROM_PARAM_CFG     = 0x13, //  获取固件参数
    EDL_COMMON_TYPE_SET_FACTORY_PARAM_CFG = 0x14,//  获取固件参数
    EDL_COMMON_TYPE_GET_FACTORY_PARAM_CFG = 0x15,//  获取固件参数
    EDL_COMMON_TYPE_RESET_FIRM_PARAM_CFG  = 0x16,//重置固件出厂设置
    EDL_COMMON_UPLOAD_GAIN_TEMPLATE   = 0x2F,   //  更新 Gain 模板
    EDL_COMMON_UPLOAD_DEFECT_TEMPLATE = 0x30,//  更新 Defect 模板
    EDL_COMMON_ERASE_FIRMWARE             = 0x4F, //  擦除固件
   EDL_COMMON_TYPE_UPDATE_BIN_FILE       = 0x50, //  更新固件
   EDL_COMMON_TYPE_SET_AQC_MODE          = 0xFF //  设置采集模式和参数
};
```

2.4.2 返回命令

```
// Notice: Response Command
enum eCallbackEventCommType
{
    ECALLBACK_TYPE_INVALVE              = 0X00,//无效命令
    ECALLBACK_TYPE_COMM_RIGHT          = 0X01,//  命令正确
    ECALLBACK_TYPE_COMM_WRONG          = 0X02,//  命令不存在
    ECALLBACK_TYPE_DUMMPLING           = 0X03,//清空命令
    ECALLBACK_TYPE_ACQ_END             = 0X04,//  连续采集结束
   ECALLBACK_TYPE_UPDATE_FIRMWARE     = 0X06,//  更新固件
   ECALLBACK_TYPE_ERASE_FIRMWARE      = 0X07,//  擦除固件
   ECALLBACK_TYPE_FPD_STATUS          = 0X09, //  状态包
    ECALLBACK_TYPE_ROM_UPLOAD          = 0X10,//  返回固件配置 ROM
    ECALLBACK_TYPE_RAM_UPLOAD          = 0X11, //  返回固件配置 RAM
    ECALLBACK_TYPE_FACTORY_UPLOAD      = 0X12,//  返回固件配置 FACTORY
    ECALLBACK_TYPE_SINGLE_IMAGE        = 0X51,//  单帧采集数据返回
    ECALLBACK_TYPE_MULTIPLE_IMAGE      = 0X52,//  连续采集数据返回
    ECALLBACK_TYPE_PACKET_MISS         = 0X5B, //  丢包数据返回
    ECALLBACK_TYPE_LIVE_ACQ_OK         = 0XA0,//  连续采集保存文件成功
    ECALLBACK_TYPE_ADMIN_CFG_UPDATA    = 0XA1,//  管理员参数
    ECALLBACK_TYPE_USER_CFG_UPDATA     = 0XA2,//  获取固件参数
    ECALLBACK_TYPE_SOCK_RECV_EVENT     = 0XA3,
    ECALLBACK_TYPE_PACKET_MISS_MSG     = 0XA4,//  单帧采集丢包，会自启动重传机制
    ECALLBACK_TYPE_THREAD_EVENT        = 0XA5,//  监控线程状态
```

```
    ECALLBACK_TYPE_CALIBRATE_ACQ_MSG   = 0XA6,//采集亮场模板素材返回消息
    ECALLBACK_TYPE_OFFSET_ERR_MSG      = 0XA7,// 生成 offset 模板成功或失败
    ECALLBACK_TYPE_GAIN_ERR_MSG        = 0XA8,// 生成 gain 模板成功或失败
    ECALLBACK_TYPE_DEFECT_ERR_MSG      = 0XA9,// 生成 defect 模板成功或失败
    ECALLBACK_TYPE_NET_ERR_MSG         = 0XAA ,// 网络异常或平板状态消息
    ECALLBACK_TYPE_SET_CFG_OK          = 0XAB // 设置 ROM 参数成功反馈
};
```

参考 HbDllType.h。

2.5 数据结构

```
// Notice:fpd software calibrate enable info.
typedef struct software_calibrate_enable_st {
    software_calibrate_enable_st(){
        enableOffsetCalib   = false;
        enableGainCalib     = false;
        enableDefectCalib   = false;
        enableDummyCalib    = false;
    };
    bool enableOffsetCalib;
    bool enableGainCalib;
    bool enableDefectCalib;
    bool enableDummyCalib;
}SOFTWARE_CALIBRATE_ENABLE;

// Notice: acq mode:static and dynamic
typedef enum
{
    STATIC_ACQ_DEFAULT_MODE = 0x00, // 默认单帧采集
    DYNAMIC_ACQ_DEFAULT_MODE,       // 默认连续采集
    DYNAMIC_ACQ_BARK_MODE,          // 创建 Offset 模板-连续采集暗场图
    DYNAMIC_ACQ_BRIGHT_MODE,        // 创建 Gain 模板-连续采集亮场图
    STATIC_ACQ_BRIGHT_MODE,         // 创建 Gain 模板-单帧采集亮场图
    STATIC_DEFECT_ACQ_MODE          // 创建 Defect 模板-缺陷校正采集亮场图
    // ADD BY MHYANG 20201013
    OFFSET_TEMPLATE_MODE,           // 连续采集一组暗场图-offset
    GAIN_TEMPLATE_MODE,             // 连续采集一组亮场图-gain
    DEFECT_ACQ_GROUP1,              // 连续采集一组亮场图-defect group1
    DEFECT_ACQ_GROUP2,              // 连续采集一组亮场图-defect group2
    DEFECT_ACQ_AND_TEMPLATE         // 连续采集一组亮场图-defect group3 and
                                    generate template
}EnumIMAGE_ACQ_MODE;

// Notice:fpd aqc mode
```

```
typedef struct fpd_aqc_mode_st {
    fpd_aqc_mode_st(){
        aqc_mode   = STATIC_ACQ_DEFAULT_MODE;
        ngroupno   = 0;
        nframesum = 0;
        ndiscard   = 0;
        nframeid   = 0;
    };
    EnumIMAGE_ACQ_MODE aqc_mode;
    int   ngroupno;
    int   nframesum;
    int   ndiscard;
    int   nframeid;
}FPD_AQC_MODE;

typedef struct ImageProperty
{
    ImageProperty() {
        nwidth     = IMAGE_WIDTH;
        nheight    = IMAGE_HEIGHT;
        datatype = 0;
        ndatabit = 0;
        nendian    = 0;
    }
    int nwidth;             //image width
    int nheight;            //image height
    int datatype;           //data type：0-unsigned char,1-char,2-unsigned short,3-float,4-double
    int ndatabit;           //data bit:0-16bit,1-14bit,2-12bit,3-8bit
    int nendian;            //endian:0-little endian,1-biger endian
}IMAGE_PROPERTY;

// Notice:fpd software calibrate enable info.
typedef struct image_correct_enable_st {
    image_correct_enable_st(){
        bFeedbackCfg = false;
        ucOffsetCorrection = 0;
        ucGainCorrection     = 0;
        ucDefectCorrection = 0;
        ucDummyCorrection    = 0;
    };
    bool bFeedbackCfg;   //true-ECALLBACK_TYPE_ROM_UPLOAD
Event,false-ECALLBACK_TYPE_SET_CFG_OK Event
    unsigned char ucOffsetCorrection;     //00-"Do nothing";01-"prepare Offset Correction";
02-"post Offset Correction";
```

```
    unsigned char ucGainCorrection;         //00-"Do nothing";01-"Software Gain Correction";
02-"Hardware Gain Correction"
    unsigned char ucDefectCorrection;     //00-"Do nothing";01-"Software Defect Correction";
02-"Software Defect Correction"
    unsigned char ucDummyCorrection;      //00-"Do nothing";01-"Software Dummy Correction";
02-"Software Dummy Correction"
}IMAGE_CORRECT_ENABLE;
```

3、API 详细说明

3.1 函数类型

API 中用到的基本数据类型如下：

void	无参数	无
void *	无符号指针	4 字节
int	有符号整型	4 字节
short	有符号短整型	2 字节
WORD	无符号短整型(unsigned short)	2 字节
char	字符型	1 字节
BOOL	有符号整型(int)	4 字节
float	单精度浮点型	4 字节
unsigned int	无符号整型	4 字节
unsigned short	无符号短整型	2 字节
int *	整型指针	4 字节
char *	字符型指针	4 字节
unsigned char	无符号字符型	1 字节
CodeStringTable	自定义类型：错误信息表	12 字节
RegCfgInfo	自定义类型：固件参数对象	1024 字节
ImgCaliCfg	自定义类型：平板图像校正参数	20 字节
SysCfgInfo	自定义类型：平板系统参数	128 字节
USER_CALLBACK_HANDLE_PROCESS	自定义类型：回调函数指针载状态	4 字节
USER_CALLBACK_HANDLE_ENVENT	自定义类型：回调函数指针	4 字节
SOFTWARE_CALIBRATE_ENABLE	自定义类型：软件校正使能	4 字节
IMAGE_PROPERTY	自定义类型：图像属性	32 字节
IMAGE_CORRECT_ENABLE	自定义类型：图像校正使能参数	5 字节
FPD_AQC_MODE	自定义类型：采集模式参数	36 字节
DOWNLOAD_T_FILE	自定义类型：下载模板参数	264 字节

CALIBRATE_INPUT_PARAM	自定义类型：校正生成模型参数	36 字节

3.2 函数列表

序号	函数名	功能
D001	HBI_Init	初始化动态库
D002	HBI_Destroy	释放动态库资源
D003	HBI_SetEnableCalibrate	设置软件校正使能状态
D004	HBI_GetEnableCalibrate	获取软件校正使能状态
D005	HBI_ConnectDetector	建立连接
D006	HBI_DisConnectDetector	断开连接
D007	HBI_RegEventCallBackFun	注册回调函数-图像、配置、状态
D008	HBI_ResetDetector	重置探测器
D009	HBI_Prepare	准备指令
D010	HBI_Dumping	清空指令
D011	HBI_SetAqcProperty	设置采集模式和参数
D012	HBI_SingleAcquisition	单帧采集
D013	HBI_LiveAcquisition	连续采集
D014	HBI_StopAcquisition	停止采集
D015	HBI_ForceStopAcquisition	强制停止采集
D016	HBI_SetRawStyle	设置图像格式和保存
D017	HBI_GetDevCfgInfo	获取固件配置信息
D018	HBI_SetSystemConfig	设置平板触发模式
D019	HBI_SetImageCalibration	设置固件校正使能状态
D020	HBI_WorkStationInitDllCfg	设置触发模式、固件校正和软件校正使能，（将 D003、D018 和 D019 合并）
D021	HBI_GetImageProperty	获取图像属性
D022	HBI_GetSDKVerion	获取 SDK Dll 版本号
D023	HBI_GetFirmareVerion	获取固件版本号
D024	HBI_GetError	暂不支持，后期维护
D025	HBI_SetGainMode	设置增益模式
D026	HBI_GetGainMode	获取增益模式
D027	HBI_SetAcqSpanTm	设置采集时间间隔
D028	HBI_GetAcqSpanTm	获取采集时间间隔
D029	HBI_SetAedIntegrateTm	设置 AED 积分时间
D030	HBI_GetAedIntegrateTm	获取 AED 积分时间
D031	HBI_QuckInitDllCfg	快速初始化参数（工作站，替换

		HBI_WorkStationInitDllCfg）旧版
D032	HBI_ResetFirmware	重置固件出厂设置
D033	HBI_TriggerAndCorrectApplay	快速初始化参数（触发模式和图像矫正使能）替换 HBI_QuckInitDllCfg
D034	HBI_UpdateTriggerMode	设置触发模式
D035	HBI_UpdateCorrectEnable	设置固件图像校正使能
D036	HBI_GetCorrectEnable	获取固件图像矫正使能
D037	HBI_SetSoftwareCorrect	设置软件校正使能，替换 HBI_SetEnableCalibrate
D038	HBI_GetSoftwareCorrect	获取软件矫正使能，替换 HBI_GetEnableCalibrate
D039	HBI_SetAedThreshold	设置 AED 阈值
D040	HBI_GetAedThreshold	获取 AED 阈值
D041	HBI_SetSaturationValue	设置图像饱和阈值
D042	HBI_GetSaturationValue	获取图像饱和阈值
D043	HBI_SetClippingValue	设置图像裁剪阈值
D044	HBI_GetClippingValue	获取图像裁剪阈值
D045	HBI_SetPreAcqTm	设置软触发单帧采集清空和采集之间的时间间隔
D046	HBI_GetPreAcqTm	获取软触发单帧采集清空和采集之间的时间间隔
D047	HBI_UploadTemplate	上传矫正模板
D048	HBI_StopUploadTemplate	停止上传矫正模板
D049	HBI_SetLPFLevel	设置低通滤波器档位参数
D050	HBI_GetLPFLevel	获取低通滤波器档位参数
D051	HBI_16UCConvertTo8UC	16 位数据转换为 8 位并自动调整窗宽窗位
D052	HBI_ImmediateGenerateTemplate	一键生成模板，包括 offset、gain 和 defect
D053	HBI_SetPreviewMode	设置 Preview 预览或取消预览
D054	HBI_SAecModeApply	设置或取消 SAEC 模式
D055	HBI_SAecAcq	SAEC 采集上图指令
...	...	...

3.3 函数

注意：

```
#ifdef _DLL_EX_IM
#define HB_PDF_DLL __declspec(dllexport)
#else
#define HB_PDF_DLL __declspec(dllimport)
#endif
#define _DLL_EX_IM 0   // 0-导入函数，1-导出函数，默认为 0（用户默认）。
```

3.3.1 HBI_Init

```
/************************************************************
* 编      号: D001
* 函 数 名：HBI_Init
* 功能描述: 初始化动态库，创建 SDK 对象指针，初始化固件信息和初始化线程消息参数。
* 参数说明：无
* 返 回 值：void *
            失败：NULL,成功：非空
* 备      注:
************************************************************/
HB_PDF_DLL void *HBI_Init();
```

3.3.2 HBI_Destroy

```
/************************************************************
* 编      号: D002
* 函 数 名：HBI_Destroy
* 功能描述: 释放动态库资源
* 参数说明：
        In: void *handle - 句柄,动态库 FPD SDK Object 对象指针
        Out: 无
* 返 回 值：void
* 备      注:
************************************************************/
HB_PDF_DLL void HBI_Destroy(void *handle);
```

3.3.3 HBI_SetEnableCalibrate

```
/************************************************************
* 编      号: D003
* 函 数 名：HBI_SetEnableCalibrate
* 功能描述: 设置校正使能
* 参数说明：
        In: void *handle - 句柄(无符号指针)
            FPD_CALIB_ENABLE inStatus - 校正使能状态，见 HbDllType.h 中
                                        FPD_CALIB_ENABLE
        Out: 无
* 返 回 值：int
        0    成功
        非 0 失败，见 HbDllError.h 错误码
* 备      注:
************************************************************/
```

字段说明：

```
    bool enableOffsetCalib; //  使能 offset
    bool enableGainCalib;//  使能 gain
    bool enableDefectCalib;//使能 defect
```

```
    bool enableDummyCalib;//使能 dummy
typedef enum
{
    INVALID_TRIGGER_MODE = 0x00,
    STATIC_SOFTWARE_TRIGGER_MODE = 0x01, //软件触发获取图像
    STATIC_PREP_TRIGGER_MODE = 0x02,
    STATIC_HVG_TRIGGER_MODE   = 0x03,
    STATIC_AED_TRIGGER_MODE   = 0x04,
    DYNAMIC_HVG_TRIGGER_MODE = 0x05,
    DYNAMIC_FPD_TRIGGER_MODE = 0x06
}EnumTRIGGER_MODE;

HB_PDF_DLL          int                 HBI_SetEnableCalibrate(void          *handle,
SOFTWARE_CALIBRATE_ENABLE inEnable);
```

3.3.4 HBI_GetEnableCalibrate

```
/*************************************************************
* 编      号: D004
* 函 数 名: HBI_GetEnableCalibrate
* 功能描述: 设置校正使能
* 参数说明:
    In: void *handle - 句柄(无符号指针)
        SOFTWARE_CALIBRATE_ENABLE inEnable - 校正使能状态,见 HbDllType.h 中
                                             SOFTWARE_CALIBRATE_ENABLE struct
    Out: 无
* 返 回 值：int
        0    成功
        非 0 失败，见 HbDllError.h 错误码
* 备      注:
*************************************************************/
HB_PDF_DLL          int          __stdcall          HBI_GetEnableCalibrate(void          *handle,
SOFTWARE_CALIBRATE_ENABLE *inEnable);
```

3.3.5 HBI_ConnectDetector

```
/*************************************************************
* 编      号: D005
* 函 数 名：HBI_ConnectDetector
* 功能描述: 建立连接
* 参数说明：
        In: void *handle - 句柄(无符号指针)
            char *szRemoteIp - 平板 IP 地址,如 192.168.10.40
            unsigned short remotePort - 平板端口,如 32897(0x8081)
            char *szlocalIp - 上位机地址,如 192.168.10.20
            unsigned short localPort -上位机端口,如 192.168.10.40(0x8080)
```

```
                int timeout - 接收超时,如 20ms
            Out: 无
* 返 回 值：int
            0    - 成功
            非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int HBI_ConnectDetector(void *handle,
                                   char *szRemoteIp,
                                   unsigned short remotePort,
                                   char *szlocalIp,
                                   unsigned short localPort,
                                   int timeout);
```

3.3.6 HBI_DisConnectDetector

```
/************************************************************
* 编      号: D006
* 函 数 名：HBI_DisConnectDetector
* 功能描述: 断开连接
* 参数说明：
            In: void *handle - 句柄(无符号指针)
            FPD_CALIB_ENABLE inStatus - 校正使能状态，见 HbDllType.h 中
FPD_CALIB_ENABLE
            Out: 无
* 返 回 值：int
                0    - 成功
                非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int HBI_DisConnectDetector(void *handle);
```

3.3.7 HBI_RegEventCallBackFun

```
/************************************************************
* 编      号: D007
* 函 数 名：HBI_RegEventCallBackFun
* 功能描述: 注册回调函数
* 参数说明：
            In: void *handle - 句柄(无符号指针)
            USER_CALLBACK_HANDLE_ENVENT callbackfun - 注册回调函数
            void *pObject - 对象指针
            Out: 无
* 返 回 值：int
            0    - 成功
            非 0 - 失败
```

```
* 备      注:
************************************************************/
HB_PDF_DLL int HBI_RegEventCallBackFun(void *handle,
USER_CALLBACK_HANDLE_ENVENT callbackfun);
```

3.3.8 HBI_ResetDetector

```
/************************************************************
* 编      号: D008
* 函 数 名：HBI_ResetDetector
* 功能描述: 重置探测器
* 参数说明：
        In: void *handle - 句柄(无符号指针)
        Out: 无
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int HBI_ResetDetector(void *handle);
```

3.3.9 HBI_Prepare

```
/************************************************************
* 编      号: D009
* 函 数 名：HBI_Prepare
* 功能描述: 准备指令
* 参数说明：
        In: void *handle - 句柄(无符号指针)
        Out: 无
* 返 回 值：int
        0    - 成功
        非 0 - 失败
0    成功
* 备      注:
************************************************************/
HB_PDF_DLL int HBI_Prepare(void *handle);
```

3.3.10 HBI_Dumping

```
/************************************************************
* 编      号: D010
* 函 数 名：HBI_Dumping
* 功能描述: 清空指令
* 参数说明：
        In: void *handle - 句柄(无符号指针)
        Out: 无
```

```
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备    注:
**********************************************************/
HB_PDF_DLL int HBI_Dumping(void *handle);
```

3.3.11 HBI_SetAqcProperty

```
/**********************************************************
* 编    号: D011
* 函 数 名：HBI_SetAqcProperty
* 功能描述: 设置采集属性
* 参数说明：
        In: void *handle - 句柄(无符号指针)
            FPD_AQC_MODE _mode - 采集模式以及参数
        Out: 无
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备    注:
**********************************************************/
HB_PDF_DLL int HBI_SetAqcProperty(void *handle, FPD_AQC_MODE _mode);
```

3.3.12 HBI_SingleAcquisition

```
/**********************************************************
* 编    号: D012
* 函 数 名：HBI_SingleAcquisition
* 功能描述: 单帧采集
* 参数说明：
        In:   void *handle - 句柄(无符号指针)
              FPD_AQC_MODE _mode - 采集模式以及参数
              其中结构体包含字段：
    EnumIMAGE_ACQ_MODE //获取图像模式包括
        STATIC_ACQ_DEFAULT_MODE = 0x00,        // 默认单帧采集
        DYNAMIC_ACQ_DEFAULT_MODE,              // 默认连续采集
        DYNAMIC_ACQ_BARK_MODE,                 // 生成模板-连续采集暗场图
        DYNAMIC_ACQ_BRIGHT_MODE,               // 生成模板-连续采集亮场图
        STATIC_ACQ_BRIGHT_MODE                 // 生成模板-单帧采集亮场图
    int   ngroupno; // 组号
    int   nframesum;//采集帧数
    int   ndiscard;//主动丢弃帧数
    int   nframeid;//帧 id 号
        Out: 无
* 返 回 值：int
```

0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int HBI_SingleAcquisition(void *handle, FPD_AQC_MODE _mode);

3.3.13 HBI_LiveAcquisition

/**********
* 编 号: D013
* 函 数 名：LiveAcquisition
* 功能描述: Live Acquisition
* 参数说明：连续采集
* 参数说明：
In: void *handle - 句柄(无符号指针)
FPD_AQC_MODE _mode - 采集模式以及参数，参考单帧
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int HBI_LiveAcquisition(void *handle, FPD_AQC_MODE _mode);

3.3.14 HBI_StopAcquisition

/**********
* 编 号: D014
* 函 数 名：StopAcquisition
* 功能描述: 停止采集
* 参数说明：
In: void *handle - 句柄(无符号指针)
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int HBI_StopAcquisition(void *handle);

3.3.15 HBI_ForceStopAcquisition

/**********
* 编 号: D015
* 函 数 名：HBI_ForceStopAcquisition
* 功能描述: 强制停止采集
* 参数说明：

```
        In: void *handle -  句柄(无符号指针)
        Out:  无
* 返 回 值：int
        0     - 成功
        非 0 -  失败
* 备      注:
************************************************************/
HB_PDF_DLL int HBI_ForceStopAcquisition(void *handle);
```

3.3.16 HBI_SetRawStyle

```
/************************************************************
* 编      号: D016
* 函 数 名：HBI_SetRawStyle
* 功能描述:  设置是否保存图像以及图像文件形式
* 参数说明：
        In: void *handle -  句柄(无符号指针)
            bool bsave -  保存拟或显示,false:显示不保存，true:保存不显示
            bool bsingleraw -  报存在单个文件或多个文件,false:1 帧数据保存 1 个文件，
true:多帧数据可保存在一个文件中
        Out:  无
* 返 回 值：int
        0     - 成功
        非 0 -  失败
* 备      注:
************************************************************/
HB_PDF_DLL int HBI_SetRawStyle(void *handle, bool bsave, bool bsingleraw = false);
```

3.3.17 HBI_GetDevCfgInfo

```
/************************************************************
* 编      号: D017
* 函 数 名: HBI_GetDevCfgInfo
* 功能描述:  获取固件的最新设备参数，同步事件
* 参数说明:
    In: void *handle -  句柄(无符号指针)
    RegCfgInfo* pRegCfg    - 设备参数结构体
    Out:  无
* 返 回 值：int
    0     - 成功
    非 0 -  失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_GetDevCfgInfo(void *handle, RegCfgInfo* pRegCfg);
```

3.3.18 HBI_SetSystemConfig

```
/************************************************************
* 编      号: D018
* 函 数 名: HBI_SetSystemConfig
* 功能描述: 设置 System Configuration Information，异步事件
* 参数说明:
    In: void *handle - 句柄(无符号指针)
    SysCfgInfo* pSysCfg  - 配置参数结构体,见 HbDllType.h 中 SysCfgInfo
    Out: 无
* 返 回 值：int
    0    - 成功
    非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_SetSystemConfig(void *handle, SysCfgInfo* pSysCfg);
```

3.3.19 HBI_SetImageCalibration

```
/************************************************************
* 编      号: D019
* 函 数 名: HBI_SetImageCalibration
* 功能描述: 设置 Image Calibration Configuration，异步事件
* 参数说明:
    In: void *handle - 句柄(无符号指针)
    ImgCaliCfg* pImgCaliCfg - 配置参数结构体,见 HbDllType.h 中 ImgCaliCfg
    Out: 无
* 返 回 值：int
    0    - 成功
    非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_SetImageCalibration(void *handle, ImgCaliCfg* pImgCaliCfg);
```

3.3.20 HBI_WorkStationInitDllCfg

```
/************************************************************
* 编      号: D020
* 函 数 名:HBI_WorkStationInitDllCfg
* 功能描述: 无线属性
* 参数说明:
    In: void *handle - 句柄(无符号指针)
    SysCfgInfo* pSysCfg- 触发模式
    ImgCaliCfg* pFirmwareCaliCfg- 固件校正使能
    SOFTWARE_CALIBRATE_ENABLE* pSoftwareCaliCfg- 软件校正使能
    Out: 无
* 返 回 值：int
```

0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_WorkStationInitDllCfg(void *handle, SysCfgInfo* pSysCfg, ImgCaliCfg* pFirmwareCaliCfg, SOFTWARE_CALIBRATE_ENABLE* pSoftwareCaliCfg);

3.3.21 HBI_GetImageProperty

/**********
* 编 号: D021
* 函 数 名: HBI_GetImageProperty
* 功能描述: 获取图像属性
* 参数说明:
In: void *handle - 句柄(无符号指针)
IMAGE_PROPERTY *img_pro - 图像属性,见 HbDllType.h 中 IMAGE_PROPERTY
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_GetImageProperty(void *handle, IMAGE_PROPERTY *img_pro);

3.3.22 HBI_GetSDKVerion

/**********
* 编 号: D022
* 函 数 名: HBI_GetSDKVerion
* 功能描述: 获取 Dll 版本号
* 参数说明:
In: void *handle - 句柄(无符号指针)
char *szVer - 版本信息字符串
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_GetSDKVerion(char *szVer);

3.3.23 HBI_GetFirmareVerion

/**********
* 编 号: D023
* 函 数 名: HBI_GetFirmareVerion

```
* 功能描述: 获取 Dll 版本号
* 参数说明:
    In: void *handle - 句柄(无符号指针)
    char *szVer  - 版本信息字符串
    Out: 无
* 返 回 值：int
    0    - 成功
    非 0 - 失败
* 备    注:
**********************************************************/
HB_PDF_DLL int __stdcall HBI_GetFirmareVerion(void *handle, char *szFirmwareVer);
```

3.3.24 HBI_GetError

```
/**********************************************************
* 编    号: No024
* 函 数 名: HBI_GetError
* 功能描述: 获取错误信息，暂不支持
* 参数说明:
    In: void *handle - 句柄(无符号指针)
        CodeStringTable* inTable - 错误信息信息列表
        int count - 信息列表个数
        int recode - 错误码
    Out:无
* 返 回 值：const char *
    非 NULL - 成功，错误信息
    NULL    - 失败
* 备    注:
**********************************************************/
HB_PDF_DLL const char * __stdcall HBI_GetError(void *handle, CodeStringTable*
inTable, int count, int recode);
```

3.3.25 HBI_SetGainMode

```
/**********************************************************
* 编    号: No025
* 函 数 名: HBI_SetGainMode
* 功能描述: 设置增益模式
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        int mode - 模式
            [n]-Invalid
            [1]-0.6pC
            [2]-1.2pC
            [3]-2.4pC
            [4]-4.8pC
```

[5]-7.2pC,默认 7.2pC
[6]-9.6pC
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_SetgGainMode(void *handle, int nGainMode);

3.3.26 HBI_GetGainMode

/**********
* 编 号: No026
* 函 数 名: HBI_GetGainMode
* 功能描述: 获取增益模式
* 参数说明:
In: void *handle - 句柄(无符号指针)
Out: 无
* 返 回 值：int
[n]-Invalid
[1]-0.6pC
[2]-1.2pC
[3]-2.4pC
[4]-4.8pC
[5]-7.2pC
[6]-9.6pC
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_GetgGainMode(void *handle);

3.3.27 HBI_SetAcqSpanTm

/**********
* 编 号: No027
* 函 数 名: HBI_SetAcqSpanTm
* 功能描述: 设置采集时间间隔
* 参数说明:
In: void *handle - 句柄(无符号指针)
int time - 间隔时间,单位是毫秒 ms，>= 1000ms
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/

```
HB_PDF_DLL int __stdcall HBI_SetAcqSpanTm(void *handle, int time);
```

3.3.28 HBI_GetAcqSpanTm

```
/************************************************************
* 编      号: No028
* 函 数 名: HBI_GetAcqSpanTm
* 功能描述: 获取采集时间间隔
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        out:int *out_time - 时间,单位是毫秒 ms
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_GetAcqSpanTm(void *handle, int *out_time);
```

3.3.29 HBI_GetAedIntegrateTm

```
/************************************************************
* 编      号: No060
* 函 数 名: HBI_SetAcqSpanTm
* 功能描述: 设置 AED 积分时间
* 参数说明:
        In: void *handle - 句柄(无符号指针)
            int time - 间隔时间,单位是毫秒 ms，>= 500ms
        Out: 无
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_SetAedIntegrateTm(void *handle, int time);
```

3.3.30 HBI_GetAedIntegrateTm

```
/************************************************************
* 编      号: No030
* 函 数 名: HBI_GetAedIntegrateTm
* 功能描述: 获取 AED 积分时间
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        out:int *out_time - 时间,单位是毫秒 ms
* 返 回 值：int
        0    - 成功
        非 0 - 失败
```

```
* 备      注:
*********************************************************/
HB_PDF_DLL int __stdcall HBI_GetAedIntegrateTm(void *handle, int *out_time);
```

3.3.31 HBI_QuckInitDllCfg

```
/*********************************************************
* 编      号: No031
* 函 数 名: HBI_QuckInitDllCfg
* 功能描述: 快速初始化参数（工作站）
* 参数说明:
        In: void *handle - 句柄(无符号指针)
            int _triggerMode,1-软触发，3-高压触发，4-FreeAED。
            ImgCaliCfg* pFirmwareCaliCfg,见 HbDllType.h。
            SOFTWARE_CALIBRATE_ENABLE* pSoftwareCaliCfg,见 HbDllType.h。
        Out:无
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:替换 HBI_WorkStationInitDllCfg 接口
*********************************************************/
HB_PDF_DLL int __stdcall HBI_QuckInitDllCfg(void *handle, int _triggerMode,
ImgCaliCfg* pFirmwareCaliCfg, SOFTWARE_CALIBRATE_ENABLE* pSoftwareCaliCfg);
```

3.3.32 HBI_ResetFirmware

```
/*********************************************************
* 编      号: No032
* 函 数 名: HBI_ResetFirmware
* 功能描述: 重置固件出厂设置
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        Out: 无
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:
*********************************************************/
HB_PDF_DLL int __stdcall HBI_ResetFirmware(void *handle);
```

3.3.33 HBI_TriggerAndCorrectApplay

```
/*********************************************************
* 编      号: No033
* 函 数 名: HBI_TriggerAndCorrectApplay
* 功能描述: 设置触发模式和图像校正使能（工作站）新版本
* 参数说明:
```

In: void *handle - 句柄(无符号指针)
int _triggerMode,1-软触发，3-高压触发，4-FreeAED。
IMAGE_CORRECT_ENABLE* pCorrect,见 HbDllType.h。
Out:无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:替换 HBI_WorkStationInitDllCfg 和 HBI_QuckInitDllCfg 接口
**********/
HB_PDF_DLL int __stdcall HBI_TriggerAndCorrectApplay(void *handle, int _triggerMode, IMAGE_CORRECT_ENABLE* pCorrect);

3.3.34 HBI_UpdateTriggerMode

/**********
* 编 号: No034
* 函 数 名: HBI_UpdateTriggerMode
* 功能描述: 设置触发模式
* 参数说明:
In: void *handle - 句柄(无符号指针)
int _triggerMode,1-软触发，3-高压触发，4-FreeAED。
Out:无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_UpdateTriggerMode(void *handle, int _triggerMode);

3.3.35 HBI_UpdateCorrectEnable

/**********
* 编 号: No035
* 函 数 名: HBI_UpdateCorrectEnable
* 功能描述: 更新图像固件校正使能
* 参数说明:
In: void *handle - 句柄(无符号指针)
IMAGE_CORRECT_ENABLE* pCorrect,见 HbDllType.h。
Out:无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_UpdateCorrectEnable(void *handle, IMAGE_CORRECT_ENABLE* pCorrect);

3.3.36 HBI_GetCorrectEnable

```
/************************************************************
* 编　　号: No036
* 函 数 名: HBI_TriggerAndCorrectApplay
* 功能描述: 获取图像固件校正使能
* 参数说明:
        In: void *handle - 句柄(无符号指针)
            IMAGE_CORRECT_ENABLE* pCorrect,见 HbDllType.h。
        Out:无
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备　　注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_GetCorrectEnable(void *handle,
IMAGE_CORRECT_ENABLE *pCorrect);
```

3.3.37 HBI_GetSoftwareCorrect

```
/************************************************************
* 编　　号: No037
* 函 数 名: HBI_SetSoftwareCorrect
* 功能描述: 设置校正使能，替换 HBI_SetEnableCalibrate
* 参数说明:
        In: void *handle - 句柄(无符号指针)
            SOFTWARE_CALIBRATE_ENABLE inEnable - 校正使能状态，见
HbDllType.h 中 SOFTWARE_CALIBRATE_ENABLE struct
        Out: 无
* 返 回 值：int
        0    成功
        非 0 失败，见 HbDllError.h 错误码
* 备　　注:新接口
************************************************************/
HB_PDF_DLL int __stdcall HBI_SetSoftwareCorrect(
void *handle,
SOFTWARE_CALIBRATE_ENABLE inEnable);
```

3.3.38 HBI_GetSoftwareCorrect

```
/************************************************************
* 编　　号: No038
* 函 数 名: HBI_GetSoftwareCorrect
* 功能描述: 获取校正使能参数，替换 HBI_GetEnableCalibrate
* 参数说明:
        In: void *handle - 句柄(无符号指针)
```

SOFTWARE_CALIBRATE_ENABLE inEnable - 校正使能状态，见HbDllType.h 中
SOFTWARE_CALIBRATE_ENABLE struct
Out: 无
* 返 回 值：int
0 成功
非 0 失败，见 HbDllError.h 错误码
* 备 注:新接口
**********/
HB_PDF_DLL int __stdcall HBI_GetSoftwareCorrect(
void *handle,
SOFTWARE_CALIBRATE_ENABLE *inEnable);

3.3.39 HBI_SetAedThreshold

/**********
* 编 号: No039
* 函 数 名: HBI_SetAedThreshold
* 功能描述: 设置 AED 阈值
* 参数说明:
In: void *handle - 句柄(无符号指针)
int out_ivalue - 阈值 [10,000~1,000,000]
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_SetAedThreshold(void *handle, int in_ivalue);

3.3.40 HBI_GetAedThreshold

/**********
* 编 号: No040
* 函 数 名: HBI_GetAedThreshold
* 功能描述: 获取 AED 阈值
* 参数说明:
In: void *handle - 句柄(无符号指针)
out:int *out_ivalue - 阈值 [10,000~1,000,000]
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_GetAedThreshold(void *handle, int *out_ivalue);

3.3.41 HBI_SetSaturationValue

```
/************************************************************
* 编      号: No041
* 函 数 名: HBI_SetSaturationValue
* 功能描述: 设置饱和值
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        out:int *out_ivalue - 饱和值 [0~65,535]
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_SetSaturationValue(void *handle, int in_ivalue);
```

3.3.42 HBI_GetSaturationValue

```
/************************************************************
* 编      号: No042
* 函 数 名: HBI_GetSaturationValue
* 功能描述: 获取饱和值
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        out:int *out_ivalue - 饱和值 [0~65,535]
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_GetSaturationValue(void *handle, int *out_ivalue);
```

3.3.43 HBI_SetClippingValue

```
/************************************************************
* 编      号: No043
* 函 数 名: HBI_SetClippingValue
* 功能描述: 设置图像剪裁值
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        out:int *out_ivalue - 剪裁值 [0~65,535]
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_SetClippingValue(void *handle, int in_ivalue);
```

3.3.44 HBI_GetClippingValue

```
/************************************************************
* 编      号: No044
* 函 数 名: HBI_GetClippingValue
* 功能描述: 获取图像剪裁值
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        out:int *out_ivalue - 剪裁值 [0~65,535]
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_GetClippingValue(void *handle, int *out_ivalue);
```

3.3.45 HBI_SetPreAcqTm

```
/************************************************************
* 编      号: No045
* 函 数 名: HBI_SetPreAcqTm
* 功能描述: 设置软触发单帧采集清空和采集之间的时间间隔
* 参数说明:
        In: void *handle - 句柄(无符号指针)
            int *in_itime - 时间 [0~10000] 单位:mm
                0-表示软触发单帧采集先 HBI_Prepare 后 HBI_SingleAcquisition 完成单帧采集
                非 0-表示软触发单帧采集，只需 HBI_Prepare 即可按照预定时间完成单帧采集
* 返 回 值：int
        0    - 成功
        非 0 - 失败
* 备      注:
************************************************************/
HB_PDF_DLL int __stdcall HBI_SetPreAcqTm(void *handle, int in_itime);
```

3.3.46 HBI_GetPreAcqTm

```
/************************************************************
* 编      号: No046
* 函 数 名: HBI_GetPreAcqTm
* 功能描述: 获取软触发单帧采集清空和采集之间的时间间隔
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        out:int *out_itime - 时间 [0~10000] 单位:mm
                0-表示软触发单帧采集先 HBI_Prepare 后 HBI_SingleAcquisition 完成单帧采集
                非 0-表示软触发单帧采集，只需 HBI_Prepare 即可按照预定时间完成单帧采集
* 返 回 值：int
```

0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_GetPreAcqTm(void *handle, int *out_itime);

3.3.47 HBI_UploadTemplate

/**********
* 编 号: No047
* 函 数 名: HBI_UploadTemplate
* 功能描述: 上传矫正模板
* 参数说明:
In: void *handle - 句柄(无符号指针)
UPLOAD_MODE *uploadmode - 上传模型对象指针
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_UploadTemplate(void *handle, UPLOAD_MODE *uploadmode);

3.3.48 HBI_StopUploadTemplate

/**********
* 编 号: No048
* 函 数 名: HBI_StopUploadTemplate
* 功能描述: 停止上传矫正模板
* 参数说明:
In: void *handle - 句柄(无符号指针)
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_StopUploadTemplate(void *handle);

3.3.49 HBI_SetLPFLevel

/**********
* 编 号: No049
* 函 数 名: HBI_SetLPFLevel
* 功能描述: 设置低通滤波器档位参数
* 参数说明:

In: void *handle - 句柄(无符号指针)
int nLPFLevel - 档位
[n]-Invalid
[0]-210khz,默认 210khz
[1]-105khz
[2]-52khz
[3]-26khz
[4]-13khz
Out: 无
* 返 回 值：int
0 - 成功
非 0 - 失败
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_SetLPFLevel(void *handle, int nLPFLevel);

3.3.50 HBI_GetLPFLevel

/**********
* 编 号: No050
* 函 数 名: HBI_GetLPFLevel
* 功能描述: 获取低通滤波器档位参数
* 参数说明:
In: void *handle - 句柄(无符号指针)
Out: 无
* 返 回 值：int
[0]-210khz,默认 210khz
[1]-105khz
[2]-52khz
[3]-26khz
[4]-13khz
[n]-Invalid
* 备 注:
**********/
HB_PDF_DLL int __stdcall HBI_GetLPFLevel(void *handle);

3.3.51 HBI_16UCConvertTo8UC

/**********
* 编 号: No051
* 功能描述: 16 位数据转换为 8 位(自动调节窗宽窗位)
* 参数说明:
In: void *handle - 句柄(无符号指针)
In/Out: unsigned char *imgbuff
In/Out: int *nbufflen
* 返 回 值：int
0 - 成功

非 0 - 失败

* 备　　注:

**********/

HB_PDF_DLL int __stdcall HBI_16UCConvertTo8UC(void *handle, unsigned char *imgbuff, int *nbufflen);

3.3.52 HBI_ImmediateGenerateTemplate

/**********

* 编　　号: No052
* 函 数 名: HBI_ImmediateGenerateTemplate
* 功能描述: 一键生成模板
* 参数说明:

In: void *handle - 句柄(无符号指针)

EnumIMAGE_ACQ_MODE _mode - 生成模板类型

OFFSET_TEMPLATE_MODE　　连续采集一组暗场图 - offset

GAIN_TEMPLATE_MODE　　连续采集一组亮场图 - gain

DEFECT_ACQ_GROUP1,　　连续采集一组亮场图 - defect group1

DEFECT_ACQ_GROUP2,　　连续采集一组亮场图 - defect group2

DEFECT_ACQ_AND_TEMPLATE 连续采集一组亮场图- defect group3 and generate template

int bprevew - 是否生成 preview 模板，1-生成，0-不生成

Out: 无

* 返 回 值：int

0　- 成功

非 0 - 失败

* 备　　注:

**********/

HB_PDF_DLL int __stdcall HBI_ImmediateGenerateTemplate(void *handle, EnumIMAGE_ACQ_MODE _mode, int bprevew = 0);

3.3.53 HBI_SetPreviewMode

/**********

* 编　　号: No053
* 函 数 名: HBI_GetPreviewMode
* 功能描述: 设置 Preview 预览或取消预览
* 参数说明:

In: void *handle - 句柄(无符号指针)

int _Mode,0-normal image,1-preview image,2-preview image and normal image。

Out:无

* 返 回 值：int

0　- 成功

非 0 - 失败

* 备　　注:

**********/

HB_PDF_DLL int __stdcall HBI_SetPreviewMode(void *handle, int inTriggMode, int inPreviewMode)

3.3.54 HBI_SAecModeApply

```
/**********
* 编　　号: No054
* 函 数 名: HBI_SAecModeApply
* 功能描述: 设置或取消 SAEC 模式
* 参数说明:
        In: void *handle - 句柄(无符号指针)
            int inPreviewMode
                0-normal image
                1-preview image
            int inPerTime  - fpd pre ready delay time
            int inPostTime - fpd post ready delay time
        Out:无
* 返 回 值：int
        0   - 成功
        非 0 - 失败
* 备　　注:分为两种情况：
          1》启用：inPreviewMode=1(preview image)，inPerTime 和 inPostTime(客户指定)
          2》取消：inPreviewMode=0(normal image)，inPerTime 和 inPostTime 默认即可
**********/
```

HB_PDF_DLL int __stdcall HBI_SAecModeApply(void *handle, int inPreviewMode, int inPerTime=0, int inPostTime=0);

3.3.55 HBI_SAecAcq

```
/**********
* 编　　号: No055
* 函 数 名: HBI_SAecAcq
* 功能描述: SAEC 采集上图指令
* 参数说明:
        In: void *handle - 句柄(无符号指针)
        Out: 无
* 返 回 值：int
        0   - 成功
        非 0 - 失败
* 备　　注:
**********/
```

HB_PDF_DLL int __stdcall HBI_SAecAcq(void *handle);

3.4 SDK 回调消息类型

回调函数类型定义：

```
// Notice: call back function
// @USER_CALLBACK_HANDLE_ENVENT
// @cmd:enum eEventCallbackCommType
// @buff
// @len
typedef int(*USER_CALLBACK_HANDLE_ENVENT)(unsigned char cmd, void *buff, int len);
```

功能描述: 回调函数定义，处理配置、图像、以及平板返回状态。

参数说明：

cmd	消息，从 SDK 返回的命令消息。
buff	返回的数据内容
len	数据值

```
// Notice: call back function
// @USER_CALLBACK_HANDLE_PROCESS
// @cmd:enum eAnswerCallbackCommType
// @buff
// @len
typedef int(*USER_CALLBACK_HANDLE_PROCESS)(unsigned char cmd, int retcode, void *inContext);
```

功能描述: 回调函数定义，处理固件更新消息反馈。

参数说明：

cmd	消息，从 SDK 返回的命令消息。
retcode	返回码
inContext	对象指针，一般为类对象指针

cmd 说明：

ECALLBACK_TYPE_INVALVE	= 0x00,返回值错误
ECALLBACK_TYPE_COMM_RIGHT	= 0x01,命令正确
ECALLBACK_TYPE_COMM_WRONG	= 0x02,命令错误
ECALLBACK_TYPE_DUMMPLING	= 0x03,删除命令
ECALLBACK_TYPE_ACQ_END	= 0x04,停止连续采集
ECALLBACK_TYPE_UPDATE_FIRMWARE	= 0X06,更细固件应答
ECALLBACK_TYPE_ERASE_FIRMWARE	= 0X07,擦除固件
ECALLBACK_TYPE_FPD_STATUS	= 0X09,FPD 状态包
ECALLBACK_TYPE_ROM_UPLOAD	= 0x10,上传 ROM 参数
ECALLBACK_TYPE_RAM_UPLOAD	= 0x11,上传 RAM 参数
ECALLBACK_TYPE_FACTORY_UPLOAD	= 0x12,上传工厂参数
ECALLBACK_TYPE_SINGLE_IMAGE	= 0x51, 单帧采集
ECALLBACK_TYPE_MULTIPLE_IMAGE	= 0x52,连续采集
ECALLBACK_TYPE_PREVIEW_IMAGE	= 0X53,Preview 图像

```
ECALLBACK_TYPE_PACKET_MISS          = 0x5B,单帧包丢失
ECALLBACK_TYPE_LIVE_ACQ_OK          = 0xA0,连续采集完成
ECALLBACK_TYPE_ADMIN_CFG_UPDATA     = 0xA1,
ECALLBACK_TYPE_USER_CFG_UPDATA      = 0xA2,
ECALLBACK_TYPE_SOCK_RECV_EVENT      = 0xA3,网络状态返回值
ECALLBACK_TYPE_PACKET_MISS_MSG      = 0xA4,丢包，值 len
ECALLBACK_TYPE_THREAD_EVENT         = 0xA5,线程状态返回，len 表示不同状态
ECALLBACK_TYPE_CALIBRATE_ACQ_MSG    = 0xA6,
ECALLBACK_TYPE_OFFSET_ERR_MSG       = 0xA7,offset 校正错误
ECALLBACK_TYPE_NET_ERR_MSG          = 0xAA,网络错误
ECALLBACK_TYPE_SET_CFG_OK           = 0XAB, //  设置 ROM 参数成功反馈
ECALLBACK_TYPE_ACQ_SUCCESS          = 0XAC, //  反馈采集到一帧图像成功
ECALLBACK_TYPE_GENERATE_TEMPLATE    = 0XAD, //  生成模板
ECALLBACK_TYPE_FILE_NOTEXIST        = 0XAE    //  文件不存在
```

4、返回码信息

详细请参考附件《HbDllError.h》。

```
    HBI_SUCCSS                          = 0,
    HBI_ERR_OPEN_DETECTOR_FAILED        = 8001,
    HBI_ERR_INVALID_PARAMS              = 8002,
    HBI_ERR_CONNECT_FAILED              = 8003,
    HBI_ERR_MALLOC_FAILED               = 8004,
    HBI_ERR_RELIMGMEM_FAILED            = 8005,
    HBI_ERR_RETIMGMEM_FAILED            = 8006,
    HBI_ERR_NODEVICE                    = 8007,
    HBI_ERR_NODEVICE_TRY_CONNECT        = 8008,
    HBI_ERR_DEVICE_BUSY                 = 8009,
    HBI_ERR_SENDDATA_FAILED             = 8010,
    HBI_ERR_RECEIVE_DATA_FAILED         = 8011,
    HBI_ERR_COMMAND_DISMATCH            = 8012,
    HBI_ERR_NO_IMAGE_RAW                = 8013,
    HBI_ERR_PTHREAD_ACTIVE_FAILED       = 8014,
    HBI_ERR_STOP_ACQUISITION            = 8015,
    HBI_ERR_INSERT_FAILED               = 8016,
    HBI_ERR_GET_CFG_FAILED              = 8017,
    HBI_NOT_SUPPORT                     = 8018,
    HBI_REGISTER_CALLBACK_FAILED        = 8019,
    HBI_SEND_MESSAGE_FAILD              = 8020,
    HBI_ERR_WORKMODE                    = 8021,
    HBI_FAILED                          = 8822,
    HBI_FILE_NOT_EXIST                  = 8823,
```

Error Message List：

```
{ 0, HBI_SUCCSS, "Success" },
{ 1, HBI_ERR_OPEN_DETECTOR_FAILED, "Open device driver failed" },
{ 2, HBI_ERR_INVALID_PARAMS, "Parameter error" },
{ 3, HBI_ERR_CONNECT_FAILED, "Connect failed" },
{ 4, HBI_ERR_MALLOC_FAILED, "Malloc memory failed" },
{ 5, HBI_ERR_RELIMGMEM_FAILED, "Releaseimagemem fail" },
{ 6, HBI_ERR_RETIMGMEM_FAILED, "ReturnImageMem fail" },
{ 7, HBI_ERR_NODEVICE, "No Device" },
{ 8, HBI_ERR_NODEVICE_TRY_CONNECT, "No Device,Try again" },
{ 9, HBI_ERR_DEVICE_BUSY, "Device busy" },
{ 10, HBI_ERR_SENDDATA_FAILED, "SendData failed" },
{ 11, HBI_ERR_RECEIVE_DATA_FAILED, "Receive Data failed" },
{ 12, HBI_ERR_COMMAND_DISMATCH, "Command dismatch" },
{ 13, HBI_ERR_NO_IMAGE_RAW, "No Image raw" },
{ 14, HBI_ERR_PTHREAD_ACTIVE_FAILED, "Pthread active failed" },
{ 15, HBI_ERR_STOP_ACQUISITION, "Pthread stop data acquisition failed" },
{ 16, HBI_ERR_INSERT_FAILED, "insert calibrate mode failed" },
{ 17, HBI_ERR_GET_CFG_FAILED, "get device config failed" },
{ 18, HBI_NOT_SUPPORT, "not surport yet" },
{ 19, HBI_REGISTER_CALLBACK_FAILED, "failed to register callback function" },
{ 20, HBI_SEND_MESSAGE_FAILD, "send message failed" },
{ 21, HBI_ERR_WORKMODE, "switch work mode failed" },
{ 22, HBI_FAILED, "operation failed" },
{ 23, HBI_FILE_NOT_EXIST, "file does not exist" }
```

详细请参考附件《HbDllError.h》。

```
/*********************************************
* 描　述：通过返回码获取当前错误信息
* 函　数：GetHbiErrorString
* 参　数：CodeStringTable* inTable  返回码表
*         int count                 返回码个数
*         int recode                返回码
* 返回值：
*         const char *  返回字符串
* 备　注: BY MH.YANG 2019/12/09
*********************************************/
const char *GetErrString(CodeStringTable* inTable, int count, int recode)
{
   const char *retString = "Unknown";
   for(int i = 0; i < count; i++) {
      if(inTable[i].HBIRETCODE == recode) {
            retString = inTable[i].errString;
            break;
```

```
    }
  }
  return retString;
}
int main()
{
    static const CodeStringTable CrErrStrList[] = {
        { 0,   HBI_SUCCSS, "Success" },
        { 1,   HBI_ERR_OPEN_DETECTOR_FAILED, "Open device driver failed" },
        { 2,   HBI_ERR_INVALID_PARAMS, "Parameter error" },
        { 3,   HBI_ERR_CONNECT_FAILED, "Connect failed" },
        { 4,   HBI_ERR_MALLOC_FAILED, "Malloc memory failed" },
        { 5,   HBI_ERR_RELIMGMEM_FAILED, "Releaseimagemem fail" },
        { 6,   HBI_ERR_RETIMGMEM_FAILED, "ReturnImageMem fail" },
        { 7,   HBI_ERR_NODEVICE, "No Device" },
        { 8,   HBI_ERR_NODEVICE_TRY_CONNECT, "No Device,Try again" },
        { 9,   HBI_ERR_DEVICE_BUSY, "Device busy" },
        { 10, HBI_ERR_SENDDATA_FAILED, "SendData failed" },
        { 11, HBI_ERR_RECEIVE_DATA_FAILED, "Receive Data failed" },
        { 12, HBI_ERR_COMMAND_DISMATCH, "Command dismatch" },
        { 13, HBI_ERR_NO_IMAGE_RAW, "No Image raw" },
        { 14, HBI_ERR_PTHREAD_ACTIVE_FAILED, "Pthread active failed" },
        { 15, HBI_ERR_STOP_ACQUISITION, "Pthread stop data acquisition failed" },
        { 16, HBI_ERR_INSERT_FAILED, "insert calibrate mode failed" },
        { 17, HBI_ERR_GET_CFG_FAILED, "get device config failed" },
        { 16, HBI_NOT_SUPPORT, "not surport yet" },
    };
  int nCode = 8001;
  const char *tmpString = GetErrString(CodeStringTable,
  sizeof(CrErrStrList)/sizeof(CodeStringTable), uValue);
  printf("nCode=%d,tmpString=%s\n", nCode, tmpString);
  return 0;
}
```

5、探测器校准

探测器校准分为偏置校准和增益校准。

偏置校准必须在关闭 X 光的情况下进行。偏置校准过程会获取一系列的暗场图像，并计算得到的这些暗场图像的平均图像。得到的平均图像用于消除探测器每个像素固有的本底偏差，即 Offset 偏置校准。

增益校准过程获取一系列无物体的曝光图像，用于消除探测器像素响应不一致所产生的图像不均匀问题。 移去曝光物体，获取不同曝光剂量下的曝光图像若干，计算并存储不同曝光剂量下的平均曝光图像，通过增益校准算法得到校准模板。

校准过程中所需的参数通过工作目录中的系统配置文件指定。以下以增益校正为例， 详细说明其工作步骤：

(1) 获取指定数量的暗场图像；
(2) 获取指定数量的不同曝光剂量下的无物体曝光图像；
(3) 根据所获取的图像生成增益图像；
(4) 最后将所生成的暗场图像、增益图像存于默认工作目录

6、开发

6.1、接口说明

-hbi_fpd_ver*.* 接口根目录
|---HBI_FPD_Tool：工具软件
|-《HBI_FPD API Programming Reference Ver*.*.pdf》接口文档
|-《factory.csv》 出厂配置导出文件，用于恢复出厂设置
|-Win32 包含 32 位 XDsicovery_Static.exe 和 HBI_FPD.dll 和 32 位 Opencv 依赖库，XML 文件位工程配置文件，可以新建、保存和修改。
|-x64 包含 64 位 XDsicovery_Static.exe 和 HBI_FPD.dll 和 64 位 Opencv 依赖库 XML 文件位工程配置文件，可以新建、保存和修改。
|--- HBI_DLL：接口文件，包含头文件、动态库 dll、静态库文件和配置文件
|- HBI_DLL\INCLUDES 头文件
|- 《HbFpdDll.h》导出函数以及说明，具体可参考接口文档。
|- 《HbDllType.h》 命令、回调函数定义返回事件类型以及固件参数结构体。
|- 《HbDllError.h》错误以及返回码信息表
|- HBI_DLL\BIN\x86 32bits 动态库 dll、静态库文件和配置文件
|- HBI_FPD.dll 和 HBI_FPD.lib 32 位 release 动态库和静态库文件
|- opencv_*** 32 位依赖库
|- HBI_DLL\BIN\x64 64bits 动态库 dll、静态库文件和配置文件
|- HBI_FPD.dll 和 HBI_FPD.lib 64 位 release 动态库和静态库文件
|- opencv_*** 64 位依赖库
|- HBI_DLL\Document 接口开发文档
|---SDK_Demo_Example：接口开发源码实例，包括静态调用程序源码,基于 vs2013
|-SDK_Demo_Example\Demo.7z 静态调用源码
|-SDK_Demo_Example\DemoDll.7z 动态调用源码

6.2、开发流程

具体见 Demo 开发程序。

6.2.1 部署

将"FPD_SDK"文件目录拷贝的用户开发指定目录下，目录下包含头文件和动态库文件、配置文件，注意 32bits 和 64bits；

6.2.2 步骤

1》用户工程中包含头文件

```
#define _DLL_EX_IM 0
#include "HbFpdDll.h" // 动态库头文件
#pragma comment(lib,"HBI_FPD.lib")
```

2》用户定义设备

```
// 设备句柄和配置
void* m_pFpd;                                    // fpd sdk 对象句柄
RegCfgInfo* m_pLastRegCfg;                       // 固件配置
FPD_AQC_MODE aqc_mode;// 采集模式和参数，单帧采集、多帧采集、暗场图以及亮场图采集

// 图像分辨率
int m_imgW, m_imgH;
char m_path[MAX_PATH];
fpd_base_cfg_st *fpd_base;
IMAGE_PROPERTY img_pro;
IMAGE_CORRECT_ENABLE* pCorrect;

// 一些基本数据类型
m_pLastRegCfg = new RegCfgInfo;
pCorrect = new IMAGE_CORRECT_ENABLE;
fpd_base = new fpd_base_cfg_st;
```

其中设备列表结构体，已在《OEMFPDSDK_TYPE.H》中定义.

3》接口调用流程

```
// 初始化 DLL 资源
void *m_pFpd = HBI_Init();
if (m_pFpd == NULL) {
   printf("HBI_Init init failed!\n");
   return false;
}

/*
* 注册回调函数
*/
// 设置注册回调函数
// myCallbackFun 回调函数
int ret = HBI_RegEventCallBackFun(m_pFpd, myCallbackFun);
if (ret != 0) {
      printf("HBI_RegEventCallBackFun failed!\n");
```

```
        HBI_Destroy(m_pFpd);
      m_pFpd = NULL;
        return false;
    }

    /*
    * 连接
    */
    ret = HBI_ConnectDetector(m_pFpd, "192.168.10.40", 32897, "192.168.10.20",
32896,20);
    if (ret != 0) {
        printf("连接失败!\n");
        HBI_Destroy(m_pFpd);
      m_pFpd = NULL;
        return false;
    }

    // 设置触发模式和图像矫正使能
    int _triggerMode = 3; // 1-软触发，3-高压触发，4-freeAED
    IMAGE_CORRECT_ENABLE* pCorrect = new IMAGE_CORRECT_ENABLE;
    if (pCorrect != NULL) {
        pcorrect->bFeedbackCfg = true;
//true-ECALLBACK_TYPE_ROM_UPLOAD
Event,false-ECALLBACK_TYPE_SET_CFG_OK Event
        pcorrect->ucOffsetCorrection = 0x02; //00-"Do nothing";01-"prepare Offset
Correction"; 02-"post Offset Correction";
        pcorrect->ucGainCorrection = 0x01; //00-"Do nothing";01-"Software Gain
Correction"; 02-"Hardware Gain Correction"
        pcorrect->ucDefectCorrection = 0x01; //00-"Do nothing";01-"Software Defect
Correction"; 02-"Software Defect Correction"
        pcorrect->ucDummyCorrection = 0x01; //00-"Do nothing";01-"Software Dummy
Correction"; 02-"Software Dummy Correction"
        ret=HBI_TriggerAndCorrectApplay(m_pFpd,_triggerMode, pCorrect);
        if (ret != 0) {
            printf("连接失败!\n");
        }
    }

    // 断开连接
    HBI_DisConnectDetector(m_pFpd));

    // 回收资源(包括断开连接和资源释放)
    HBI_Destroy(m_pFpd);
    m_pFpd = NULL;
```

//一键生成或者分布生成模板接口见【7 校正】和 【8 一键生成模板】

注意：

1》连接成功后，平板会自动反馈 ROM 参数。

2》HBI_Init 和 HBI_Destroy：连接和断开平板对应；

3》HBI_ConnectDetector 和 HBI_DisConnectDetector：初始化和释放设备对应；

4》HBI_SetSystemConfig 和 HBI_GetLastestConfig：都是读取平板固件参数

HBI_SetSystemConfig 是向固件发请求获取参数，异步函数；HBI_GetLastestConfig 是连接成功或设置成功后获取参数，同步函数。

5》HBI_ConnectDetector 连接平板，回调事件 ECALLBACK_TYPE_ROM_UPLOAD 反馈当前固件的参数，这里基本信息已固化好，用户可以直接使用

6》HBI_TriggerAndCorrectApplay，根据参数反馈 **ECALLBACK_TYPE_ROM_UPLOAD** 反馈当前固件的参数和 **ECALLBACK_TYPE_SET_CFG_OK** 事件确认成功，用户根据实际情况设置参数。

7》HBI_Destroy 释放资源，句柄为 NULL，如果直接关闭，调用 HBI_Destroy 即可，HBI_Destroy 中已包含 HBI_DisConnectDetector 的调用。

8》定义回调函数

```
##############################################
#回调函数及事件说明：
#command：事件 ID
#buff:缓冲区地址
#len:缓冲区长度
#nid：图像数据为帧号
##############################################
int CDemoDlg::myCallbackFun(unsigned char command, void *buff, int len, int nId)
{
    int ret = 0;
    if ((command == ECALLBACK_TYPE_SINGLE_IMAGE) ||
        (command == ECALLBACK_TYPE_MULTIPLE_IMAGE) ||
        (command == ECALLBACK_TYPE_ROM_UPLOAD)) {
        if (buff == NULL || len == 0) {
            printf("注册回调函数参数异常!\n");
            return ret;
        }
    }

    int status = -1;
    int j = 0;
    ret = 1;
    switch (command)
    {
    case ECALLBACK_TYPE_NET_ERR_MSG: // 平板状态：连接/断开/ready/busy
        printf("ECALLBACK_TYPE_NET_ERR_MSG,recode=%d\n", len);
        }
        break;
```

```
    case ECALLBACK_TYPE_ROM_UPLOAD:/* 更新配置 */
        printf("OEMFPD_CALLBACK_EVENT_ROM_CONFIG:\n");

        break;
    case ECALLBACK_TYPE_SET_CFG_OK:
        printf("ECALLBACK_TYPE_SET_CFG_OK:\n");
        break;
    case ECALLBACK_TYPE_SINGLE_IMAGE:    // 单帧采集上图
    case ECALLBACK_TYPE_MULTIPLE_IMAGE:  // 连续采集上图
        break;
    case ECALLBACK_TYPE_THREAD_EVENT:
    default:
        printf("ECALLBACK_TYPE_THREAD_EVENT,command=%02x\n", command);
        break;
    }
    return ret;
}
```

通过回调函数返回，图像的属性，长宽以及位数、类型以及大小端信息、触发模式、固件校正使能（目前只支持 offset 校正，后续会增加其他校正）等。

通过回调函数返回图像数据地址以及长度，如果需要器参数请说明。

6.2.3 问题和建议

1、多次出现 socket 异常，错误码是 10049 和 100048，

具体原因：1>端口被占用，比如已经打开一个程序，尝试打开第二程序会报 10049 错误码

2>关闭 socket 资源没有被释放完全再次连接导致，报 10048 错误码。

解决办法：

避免打开多个程序占用相同的端口；

确定断开失败后再连接。

2、Socket 异常，发送数据失败，错误码 10051

具体原因：

本地网络地址不正确；

解决办法：

检查配置并修改。

3、出现 socket 异常，错误码是 10040

具体原因：接收缓冲区溢出或空间不足，原因是固件为 jumbo 版本而软件采用的是非 jumbo 版本，固件和软件版本不匹配。

解决办法：

根据客户的要求，替换固件或软件版本；

4、Release 和 debug 动态库和静态库文件有区别

最近个别用户用 release 库文件开发时反馈自己在调试模式下出现崩溃现象，根据现象

排查验证，

具体原因：

用户的 release 库和用户的 debug 环境不兼容导致，目前提供给用户为 release 版本;

解决办法：

如果有需要联系研发人员获取 debug 版库文件。

5、个别用户开发工程中出现固件参数混乱

具体原因：

用户设置参数时，固件参数与上位机参数未同步，下发参数时将异常参数下发导致参数混论；

解决办法：

目前已处理，为了用户方便开发，提供对应的接口函数，API 函数内部自动同步，用户只需修改部分参数即可。

见 Demo 开发程序。

经过排查日志，发现的一些问题，建议：

1、多次出现 socket 异常，错误码是 10049 和 100048

具体原因：1>端口被占用或 Ip 地址错误，比如已经打开一个程序，尝试打开第二个程序连接会时报 10049 错误码

2>关闭 socket 资源没有被释放完全再次连接导致，报 10048 错误码。

解决办法：

避免打开多个程序占用相同的端口；

确定断开失败后再连接。

2、出现 socket 异常，错误码是 10040

具体原因：接收缓冲区溢出或空间不足，原因是固件为 jumbo 版本而软件采用的是非 jumbo 版本，固件和软件版本不匹配。

解决办法：

根据客户的要求，替换固件或软件版本；

3、Socket 异常，发送数据失败，错误码 10051

具体原因：

本地网络地址不正确；

解决办法：

检查配置并修改。

4、Release 和 debug 动态库和静态库文件有区别

最近个别用户用 release 库文件开发时反馈自己在调试模式下出现崩溃现象，根据现象排查验证，

具体原因：

用户的 release 库和用户的 debug 环境不兼容导致，目前提供给用户为 release 版本;

解决办法：

如果有需要联系研发人员获取 debug 版库文件。

5、连接成功，以平板成功反馈 ROM 参数视为与平板通讯成功。

6、调用 HBI_QuckInitDllCfg 之后等平板参数发聩后再 HBI_GetDevCfgInfo，否则出现修改的参数未被同步，

因为我们目前只有平板正常反馈后认为参数设置成功。

7、日志反馈的流程基本正常，但注意控制过程，简化调用流程，如果需要反馈固件参数请联系昊博研发人员。

8、回调函数使用

以 vs 为例，支持静态全局函数或静态成员函数，个别用户集成中出现指针访问的问题

7、分布生成校正模板

校正目前分为：offset 校正、gain 校正和 defect 校正。

1》目前固件只支持 offset 校正，软件支持 offset、gain 和 defect 校正，默认是固件 offset 校正，软件做 gain 和 defect 校正；

2》如果固件做 offset 校正，那么软件只需要做 gain 和 defect 模板即可；

3》如果软件做 offset 校正，那么 gain 和 defect 模板依赖 offset 模板；

4》固件校正与软件校正互斥关系。

5》利用模板向导生成模板后，将模板文件夹放在“HBI_FPD.dll”同目录下，启用校正使能将会采图后自动加载模板实现校正。

```
//==========================================
// 校正接口对接解决方案
//==========================================
过程：初始化模型-》采图-》导入校正模型数据-》生成模板
//
// Notice:fpd aqc mode
typedef struct fpd_aqc_mode_st {
    fpd_aqc_mode_st(){
        aqc_mode    = STATIC_ACQ_DEFAULT_MODE;
        ngroupno    = 0;
        nframesum = 0;
        ndiscard    = 0;
        nframeid    = 0;
    };
    EnumIMAGE_ACQ_MODE aqc_mode; // 采集模式
    int    ngroupno;                    // 组号，gain 或 defect 采集
    int    nframesum;                   // 连续采集:每组的总帧数，单帧默认为 0
    int    ndiscard;                   // 连续采集:忽略前几帧，单帧默认为 0
    int    nframeid;                    // 连续采集:帧号
}FPD_AQC_MODE;
```

```
// Notice:generate calibrate template input param
typedef struct calibrate_input_param_st {
    calibrate_input_param_st()
    {
        image_w          = 0;
        image_h          = 0;
        roi_x            = 0;
        roi_y            = 0;
        roi_w            = 0;
        roi_h            = 0;
        group_sum        = 0;
        per_group_num = 0;
    }
    // 图像分辨率，例如 4343 的平板：3072 * 3072，3543 板子，2560*3072
    int image_w;          // image width  图像的宽  3072
    int image_h;          // image height 图像的高  3072
    // 有效数据起始位置、宽和高，例如 3543 板子的有效区域(60, 0, 2500, 3052)，4343 的板子为(0, 0, 3072, 3072)
    int roi_x;            // ROI left      60
    int roi_y;            // ROI top       0
    int roi_w;            // ROI width     2500
    int roi_h;            // ROI height    3052
    int group_sum;        // group sum        总组数
    int per_group_num; // num per group  每一组包含的文件个数
}CALIBRATE_INPUT_PARAM;
```

7.1 offset 校正流程

7.1.1 开始连续采集

```
FPD_AQC_MODE _mode；
    _mode.aqc_mode = DYNAMIC_ACQ_BARK_MODE; // 采集暗场图
    _mode.nframesum = 20; // 采集总帧数 20 张
    _mode.nframeid =   0;   // 帧号为 0
    _mode.ngroupno =   0;   // 组号为 0
    _mode.ndiscard =   5;   // 前 5 帧抛弃
HBI_LiveAcquisition(void *handle, FPD_AQC_MODE _mode);
```

7.1.2 初始化生成 offset 校正模板的模型

```
    CALIBRATE_INPUT_PARAM offset_calibrate_param;
    offset_calibrate_param.image_w          = 3072;
    offset_calibrate_param.image_h          = 3072;
    offset_calibrate_param.roi_x            = 0;
    offset_calibrate_param.roi_y            = 0;
```

offset_calibrate_param.roi_w = 3072;
offset_calibrate_param.roi_h = 3072;
offset_calibrate_param.group_sum = 1;
offset_calibrate_param.per_group_num = 15; // 默认采集 20 帧，保存 15 帧，前 5 帧抛弃
HBI_InitOffsetMode(void *handle, CALIBRATE_INPUT_PARAM offset_calibrate_param)

7.1.3 将采集到的数据导入模型

例如，默认时 15 帧，将 15 帧图像数据分别插入到该校正模型
int group_id; // offset 链表序号
char *filepath; // 暗场文件绝对路径，如：E:****.raw
HBI_InsertOffsetMode(void *handle, int group_id, char *filepath)

例如：

```
for (int i=0; i<15; i++)
    HBI_InsertOffsetMode(handle, 0, filepath)
```

7.1.4 生成 offset 模板

int raw_num = 15; // 实际插入模型数据总个数
int bprevew - 是否生成 preview 模板，1-生成，0-不生成
HBI_GenerateOffsetTemp(void *handle, int raw_num, int bprevew = 0);

7.2 gain 校正流程

较 offset 生成模板过程，gain 和 defec 相对复杂，gain 和 defec 过程类似。
目前以 3 组每组 3 帧为例。

7.2.1 开始采集

可分为单帧采集和连续采集。

7.2.1.1 单帧采集

即曝光一次采集一帧，一般可选择软触发、高压(HVG)触发和 FreeAED 模式三种模式
1》软触发：软件发送 prepare 清空命令后，高压曝光后，发送单帧采集命令，完成一帧采集

FPD_AQC_MODE aqc_mode;
第一组(第一帧):
清空命令
HBI_Prepare(m_pFpd)
单帧采集命令
aqc_mode.aqc_mode = STATIC_ACQ_BRIGHT_MODE;
aqc_mode.nframesum = 3; // 每组采集 3 张
aqc_mode.nframeid = 0; // 帧号（0~2）
aqc_mode.ngroupno = 0; // 组号（0~2）
aqc_mode.ndiscard = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集

第一组(第二帧):

清空命令

HBI_Prepare(m_pFpd)

单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_ACQ_BRIGHT_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 1;    // 帧号（0~2）
aqc_mode.ngroupno    = 0;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第一组(第三帧):

清空命令

HBI_Prepare(m_pFpd)

单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_ACQ_BRIGHT_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 2;    // 帧号（0~2）
aqc_mode.ngroupno    = 0;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第二组(第一帧):

清空命令

HBI_Prepare(m_pFpd)

单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_ACQ_BRIGHT_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 0;    // 帧号（0~2）
aqc_mode.ngroupno    = 1;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第二组(第二帧):

清空命令

HBI_Prepare(m_pFpd)

单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_ACQ_BRIGHT_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 1;    // 帧号（0~2）
aqc_mode.ngroupno    = 1;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第二组(第三帧):

清空命令

HBI_Prepare(m_pFpd)

单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_ACQ_BRIGHT_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 2;    // 帧号（0~2）
aqc_mode.ngroupno    = 1;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第三组(第一帧):

清空命令

HBI_Prepare(m_pFpd)

单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_ACQ_BRIGHT_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 0;    // 帧号（0~2）
aqc_mode.ngroupno    = 2;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第三组(第二帧):

清空命令

HBI_Prepare(m_pFpd)

单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_ACQ_BRIGHT_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 1;    // 帧号（0~2）
aqc_mode.ngroupno    = 2;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第三组(第三帧):

清空命令

HBI_Prepare(m_pFpd)

单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_ACQ_BRIGHT_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 2;    // 帧号（0~2）
aqc_mode.ngroupno    = 2;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

直到采集 3 组每组 3 帧，一共 9 帧数据完成。

2》高压触发：用户通过按手闸控制曝光后自动上图。

```
FPD_AQC_MODE _mode；
第一组：
_mode.aqc_mode = STATIC_ACQ_BRIGHT_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   0;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);

第二组：
_mode.aqc_mode = STATIC_ACQ_BRIGHT_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   1;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);

第三组：
_mode.aqc_mode = STATIC_ACQ_BRIGHT_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   2;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);
每次采完一组后，重新设置采集属性
```

3》FreeAED：用户通过按手闸控制曝光后自动上图或者有 X 光照射后自动上图

```
FPD_AQC_MODE _mode；
第一组：
_mode.aqc_mode = STATIC_ACQ_BRIGHT_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   0;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);

第二组：
_mode.aqc_mode = STATIC_ACQ_BRIGHT_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   1;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
```

HBI_SetAqcProperty(m_pFpd, _mode);

第三组：

_mode.aqc_mode = STATIC_ACQ_BRIGHT_MODE; // 采集亮场图

_mode.nframesum = 3; // 每组采集 3 张

_mode.nframeid = 0; // 帧号（0~2）

_mode.ngroupno = 2; // 组号（0~2）

_mode.ndiscard = 0; // 单帧默认为 0

HBI_SetAqcProperty(m_pFpd, _mode);

每次采完一组后，重新设置采集属性

7.2.1.2 连续采集

持续曝光后，触发连续采集，一般选择软触发模式和高压(HVG)触发模式

1》软触发模式

FPD_AQC_MODE _mode；

第一组：

连续采集命令

_mode.aqc_mode = DYNAMIC_ACQ_BRIGHT_MODE; // 采集亮场图

_mode.nframesum = 6; // 采集总帧数 6 张

_mode.nframeid = 0; // 默认为 0

_mode.ngroupno = 0; // 组号（0~2），从 0 开始

_mode.ndiscard = 3; // 前 3 帧抛弃

HBI_LiveAcquisition(m_pFpd, aqc_mode);

第二组：

连续采集命令

_mode.aqc_mode = DYNAMIC_ACQ_BRIGHT_MODE; // 采集亮场图

_mode.nframesum = 6; // 采集总帧数 6 张

_mode.nframeid = 0; // 默认为 0

_mode.ngroupno = 1; // 组号 1

_mode.ndiscard = 3; // 前 3 帧抛弃

HBI_LiveAcquisition(m_pFpd, aqc_mode);

第三组：

连续采集命令

_mode.aqc_mode = DYNAMIC_ACQ_BRIGHT_MODE; // 采集亮场图

_mode.nframesum = 6; // 采集总帧数 6 张

_mode.nframeid = 0; // 默认为 0

_mode.ngroupno = 2; // 组号 2

_mode.ndiscard = 3; // 前 3 帧抛弃

HBI_LiveAcquisition(m_pFpd, aqc_mode);

2》高压(HVG)触发模式

高压触发和软触发模式的连续采集类似，如上。

7.2.2 初始化生成 gaint 校正模板的模型

```
CALIBRATE_INPUT_PARAM gain_calibrate_param;
gain_calibrate_param.image_w          = 3072;
gain_calibrate_param.image_h          = 3072;
gain_calibrate_param.roi_x            = 0;
gain_calibrate_param.roi_y            = 0;
gain_calibrate_param.roi_w            = 3072;
gain_calibrate_param.roi_h            = 3072;
gain_calibrate_param.group_sum        = 3;
gain_calibrate_param.per_group_num = 3; // 采集 3 组，每组 3 帧，共 9 帧
HBI_InitGainMode(void *handle, CALIBRATE_INPUT_PARAM gain_calibrate_param);
```

7.2.3 将采集到的数据导入模型

```
void *handle;      // dll 句柄
int group_id;      // 组 ID
char *filepath;    // 每组文件的句对路径
  HBI_InsertGainMode(void *handle, int group_id, char *filepath);

  将数据按照组导入到 gain 校正模板模型中：
  int m_gainGroupSum = 3;
  int nPerGroup = 3;
  for (int i = 0; i < m_gainGroupSum; i++) {
      for (unsigned int j = 0; j < nPerGroup; j++) {
          //if (0 != theDoc->Insert2GainMode(i, pGroup->raw_file[j]->filename)) {
          //    success = false;
          //    break;
          //}
          HBI_InsertGainMode(m_pFpd, i, filename[j])
      }
  }
```

7.2.4 生成 gain 模板

```
void *handle;          // dll 句柄
int group_sum;         // 组数，3 组
int per_group_num; // 每组 3 帧图像
int bprevew -  是否生成 preview 模板，1-生成，0-不生成
HBI_GenerateGainTemp(void *handle, int group_sum, int per_group_num, int bprevew = 0);
```

7.3 defect 校正流程

defect 过程与 gain 类似，下面以 3 组每组 3 帧为例。

7.3.1 开始采集

可分为单帧采集和连续采集。

7.3.1.1 单帧采集

即曝光一次采集一帧，一般可选择软触发、高压(HVG)触发和 FreeAED 模式三种模式

1》软触发：软件发送 prepare 清空命令后，高压曝光后，发送单帧采集命令，完成一帧采集

```
FPD_AQC_MODE aqc_mode;
第一组(第一帧):
清空命令
HBI_Prepare(m_pFpd)
单帧采集命令
aqc_mode.aqc_mode    = STATIC_DEFECT_ACQ_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 0;    // 帧号（0~2）
aqc_mode.ngroupno    = 0;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集

第一组(第二帧):
清空命令
HBI_Prepare(m_pFpd)
单帧采集命令
aqc_mode.aqc_mode    = STATIC_DEFECT_ACQ_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 1;    // 帧号（0~2）
aqc_mode.ngroupno    = 0;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集

第一组(第三帧):
清空命令
HBI_Prepare(m_pFpd)
单帧采集命令
aqc_mode.aqc_mode    = STATIC_DEFECT_ACQ_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
aqc_mode.nframeid    = 2;    // 帧号（0~2）
aqc_mode.ngroupno    = 0;    // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集

第二组(第一帧):
清空命令
HBI_Prepare(m_pFpd)
单帧采集命令
aqc_mode.aqc_mode    = STATIC_DEFECT_ACQ_MODE;
aqc_mode.nframesum = 3;    // 每组采集 3 张
```

```
aqc_mode.nframeid    = 0;   // 帧号（0~2）
aqc_mode.ngroupno    = 1;   // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第二组(第二帧):
清空命令
HBI_Prepare(m_pFpd)
单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_DEFECT_ACQ_MODE;
aqc_mode.nframesum = 3;   // 每组采集 3 张
aqc_mode.nframeid    = 1;   // 帧号（0~2）
aqc_mode.ngroupno    = 1;   // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第二组(第三帧):
清空命令
HBI_Prepare(m_pFpd)
单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_DEFECT_ACQ_MODE;
aqc_mode.nframesum = 3;   // 每组采集 3 张
aqc_mode.nframeid    = 2;   // 帧号（0~2）
aqc_mode.ngroupno    = 1;   // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第三组(第一帧):
清空命令
HBI_Prepare(m_pFpd)
单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_DEFECT_ACQ_MODE;
aqc_mode.nframesum = 3;   // 每组采集 3 张
aqc_mode.nframeid    = 0;   // 帧号（0~2）
aqc_mode.ngroupno    = 2;   // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第三组(第二帧):
清空命令
HBI_Prepare(m_pFpd)
单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_DEFECT_ACQ_MODE;
aqc_mode.nframesum = 3;   // 每组采集 3 张
```

```
aqc_mode.nframeid    = 1;   // 帧号（0~2）
aqc_mode.ngroupno    = 2;   // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

第三组(第三帧):

清空命令

```
HBI_Prepare(m_pFpd)
```

单帧采集命令

```
aqc_mode.aqc_mode    = STATIC_DEFECT_ACQ_MODE;
aqc_mode.nframesum = 3;   // 每组采集 3 张
aqc_mode.nframeid    = 2;   // 帧号（0~2）
aqc_mode.ngroupno    = 2;   // 组号（0~2）
aqc_mode.ndiscard    = 0;
HBI_SingleAcquisition(m_pFpd, aqc_mode); // 每次采集
```

直到采集 3 组每组 3 帧，一共 9 帧数据完成。

2》高压触发：用户通过按手闸控制曝光后自动上图。

```
FPD_AQC_MODE _mode;
```

第一组：

```
_mode.aqc_mode = STATIC_DEFECT_ACQ_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   0;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);
```

第二组：

```
_mode.aqc_mode = STATIC_DEFECT_ACQ_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   1;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);
```

第三组：

```
_mode.aqc_mode = STATIC_DEFECT_ACQ_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   2;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);
```

每次采完一组后，重新设置采集属性

3》FreeAED：用户通过按手闸控制曝光后自动上图或者有 X 光照射后自动上图

```
FPD_AQC_MODE _mode;
第一组：
_mode.aqc_mode = STATIC_DEFECT_ACQ_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   0;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);

第二组：
_mode.aqc_mode = STATIC_DEFECT_ACQ_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   1;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);

第三组：
_mode.aqc_mode = STATIC_DEFECT_ACQ_MODE; // 采集亮场图
_mode.nframesum = 3;   // 每组采集 3 张
_mode.nframeid =   0;   // 帧号（0~2）
_mode.ngroupno =   2;   // 组号（0~2）
_mode.ndiscard =   0;   // 单帧默认为 0
HBI_SetAqcProperty(m_pFpd, _mode);
每次采完一组后，重新设置采集属性
```

7.3.1.2 连续采集

持续曝光后，触发连续采集，一般选择软触发模式和高压(HVG)触发模式

1》软触发模式

```
FPD_AQC_MODE _mode;
第一组：
连续采集命令
  _mode.aqc_mode = DYNAMIC_DEFECT_ACQ_MODE; // 采集亮场图
  _mode.nframesum = 6;   // 采集总帧数 6 张
  _mode.nframeid =   0;   // 默认为 0
  _mode.ngroupno =   0;   // 组号（0~2），从 0 开始
  _mode.ndiscard =   3;   // 前 3 帧抛弃
  HBI_LiveAcquisition(m_pFpd, aqc_mode);

  第二组：
连续采集命令
  _mode.aqc_mode = DYNAMIC_DEFECT_ACQ_MODE; // 采集亮场图
```

```
_mode.nframesum = 6;   // 采集总帧数 6 张
_mode.nframeid =   0;   // 默认为 0
_mode.ngroupno =   1;   // 组号 1
_mode.ndiscard =   3;   // 前 3 帧抛弃
HBI_LiveAcquisition(m_pFpd, aqc_mode);
```

第三组：

连续采集命令

```
_mode.aqc_mode = DYNAMIC_DEFECT_ACQ_MODE; // 采集亮场图
_mode.nframesum = 6;   // 采集总帧数 6 张
_mode.nframeid =   0;   // 默认为 0
_mode.ngroupno =   2;   // 组号 2
_mode.ndiscard =   3;   // 前 3 帧抛弃
HBI_LiveAcquisition(m_pFpd, aqc_mode);
```

2》高压(HVG)触发模式

高压触发和软触发模式的连续采集类似，如上。

7.3.2 初始化生成 defect 校正模板的模型

```
CALIBRATE_INPUT_PARAM defect_calibrate_param;
defect_calibrate_param.image_w          = 3072;
defect_calibrate_param.image_h          = 3072;
defect_calibrate_param.roi_x            = 0;
defect_calibrate_param.roi_y            = 0;
defect_calibrate_param.roi_w            = 3072;
defect_calibrate_param.roi_h            = 3072;
defect_calibrate_param.group_sum        = 3;
defect_calibrate_param.per_group_num = 3; // 采集 3 组，每组 3 帧，共 9 帧
HBI_InitDefectMode(void *handle, CALIBRATE_INPUT_PARAM defect_calibrate_param);
```

7.3.3 将采集到的数据导入模型

```
void *handle;      // dll 句柄
int group_id;      // 组 ID
char *filepath;   // 每组文件的句对路径
HBI_InsertDefectMode(void *handle, int group_id, char *filepath);
```

将数据按照组导入到 gain 校正模板模型中：

```
int m_defectGroupSum = 3;
int nPerGroup = 3;
for (int i = 0; i < m_defectGroupSum; i++) {
    for (unsigned int j = 0; j < nPerGroup; j++) {
        //if (0 != theDoc->Insert2DefectMode(i, pGroup->raw_file[j]->filename)) {
        //    success = false;
        //    break;
```

```
            //}
            HBI_InsertDefectMode(m_pFpd, i, filename[j])
        }
    }
```

7.3.4 生成 defect 模板

```
void *handle;          // dll 句柄
int group_sum;         // 组数，3 组
int per_group_num; // 每组 3 帧图像
HBI_GenerateDefectTemp(void *handle, int group_sum, int per_group_num);
```

7.4 关闭向导

注意：完成模板制作后，注意切换模式，切记、切记、切记！！！，否则导致 AED 或高压触发不能触发回调上图给用户。

```
FPD_AQC_MODE _mode；
    _mode.aqc_mode = STATIC_ACQ_DEFAULT_MODE; // 默认单帧采集
    _mode.nframesum = 0;    // 默认 0
    _mode.nframeid =    0;     // 默认 0
    _mode.ngroupno =    0;    // 默认 0
    _mode.ndiscard =    0;     // 默认 0
HBI_SetAqcProperty(m_pFpd, _mode)
```

7.5 校正使能

设置软件校正使能

```
HBI_TriggerAndCorrectApplay(void *handle, int _triggerMode, IMAGE_CORRECT_ENABLE* pCorrect);
```

固件校正与软件校正互斥关系，设置参数时应注意。具体见 HBI_TriggerAndCorrectApplay 说明。

8、一键生成模板

生成模板功能快速集成，简化集成开发难度，满足不同开发用户需求，特开发此接口。下面我们将对 offset、gain 和 defect 的生成过程做简单说明。

注意：采集亮场图时不能有物料遮挡等，即“空场”，其次保证光源充分覆盖，否则出现“毛边”现象。因为亮场每组采集的帧数固定，保证高压持续稳定曝光很关键。

8.1 offset 模板

采集 1 组暗场图，直到返回成功，否则返回失败。

```
EnumIMAGE_ACQ_MODE enumTemplateType=EnumIMAGE_ACQ_MODE::DEFECT_ACQ_GROUP1;
int ret = HBI_ImmediateGenerateTemplate(theDoc->m_pFpd, enumTemplateType);
if (ret != HBI_SUCCSS) {}// 失败
else {} //成功
```

8.2 Gain 模板

采集 1 组亮场图，整常高压，毫安秒正常，调用接口前打开高压

```
EnumIMAGE_ACQ_MODE enumTemplateType = EnumIMAGE_ACQ_MODE::DEFECT_ACQ_GROUP1;
int ret = HBI_ImmediateGenerateTemplate(theDoc->m_pFpd, enumTemplateType);
if (ret != HBI_SUCCSS) {}// 失败
else {} //成功
```

8.3 Defect 模板

采集 3 组亮场图

1》正常高压，毫安秒调节正常的 10%

```
EnumIMAGE_ACQ_MODE enumTemplateType = EnumIMAGE_ACQ_MODE::DEFECT_ACQ_GROUP1;
int ret =HBI_ImmediateGenerateTemplate(theDoc->m_pFpd, enumTemplateType);
if (ret != HBI_SUCCSS) {}// 失败
else {} //成功
```

2》正常高压，毫安秒调节正常的 50%

```
EnumIMAGE_ACQ_MODE enumTemplateType = EnumIMAGE_ACQ_MODE::DEFECT_ACQ_GROUP2;
int ret =HBI_ImmediateGenerateTemplate(theDoc->m_pFpd, enumTemplateType);
if (ret != HBI_SUCCSS) {}// 失败
else {} //成功
```

3》正常高压，毫安秒调节正常

```
EnumIMAGE_ACQ_MODE enumTemplateType = EnumIMAGE_ACQ_MODE::DEFECT_ACQ_AND_TEMPLATE;
int ret =HBI_ImmediateGenerateTemplate(theDoc->m_pFpd, enumTemplateType);
if (ret != HBI_SUCCSS) {}// 失败
else {} //成功
```

8.4 校正使能

```
IMAGE_CORRECT_ENABLE *pcorrect = new IMAGE_CORRECT_ENABLE;
if (pcorrect != NULL)  {
        pcorrect->bFeedbackCfg = false;
        pcorrect->ucOffsetCorrection     = 0x01; // 0x00-不做校正，0x01-软件 offset，0x02-固件 offset
        pcorrect->ucGainCorrection      = 0x01;// 0x00-不做校正，0x01-软件 gain，0x02-固件暂不支持
        pcorrect->ucDefectCorrection      = 0x01;// 0x00-不做校正，0x01-软件 defect，0x02-固件暂不支持
       pcorrect->ucDummyCorrection   = 0x00;// 0x00-不做校正，0x01-暂不支持，0x02-固件暂不支持
        ret = theDoc->UserSetImageCorrect(pcorrect);
        if (ret == 0)
              // 成功
        else
              // 失败
       // 释放资源
        delete pcorrect;
        pcorrect = NULL;
}
```

9、Preview 模式

通过对原数据按照一定的规律(目前是 4*4 即 16 个点抽一个点)采点组成新的数据矩阵并上传给上位机(例如，工作站)，便于上位机软件通过对 Preview 数据分析调整当前高压剂量。

Preview 数据的分辨率默认为原数据的 1/16(即 width / 4,height / 4)。

重要接口函数：HBI_SetPreviewMode(void *handle, int inMode)

其中通过_Mode 来设置当前图像模式,0-正常图像,1-Preview 图像,2-暂不支持。

流程：

1、inMode=1，调用 HBI_SetPreviewMode，设置为 Preview 模式，设置成功后回调 ECALLBACK_TYPE_ROM_UPLOAD 事件反馈更新参数，

m_sImageHeight 和 m_sImageHeight 将会被更新为最新的大小，分别为原值的 1/4；

2、软触发或高压或者 AED 触发发起采集消息，将会通过 ECALLBACK_TYPE_PREVIEW_IMAGE 自动上传 Preview 图像；

3、处理完 Preview 图像结束后取消 Preview 模式，即 inMode=0，调用 HBI_SetPreviewMode 即可。m_sImageHeight 和 m_sImageHeight 也将会被更新为实际大小；

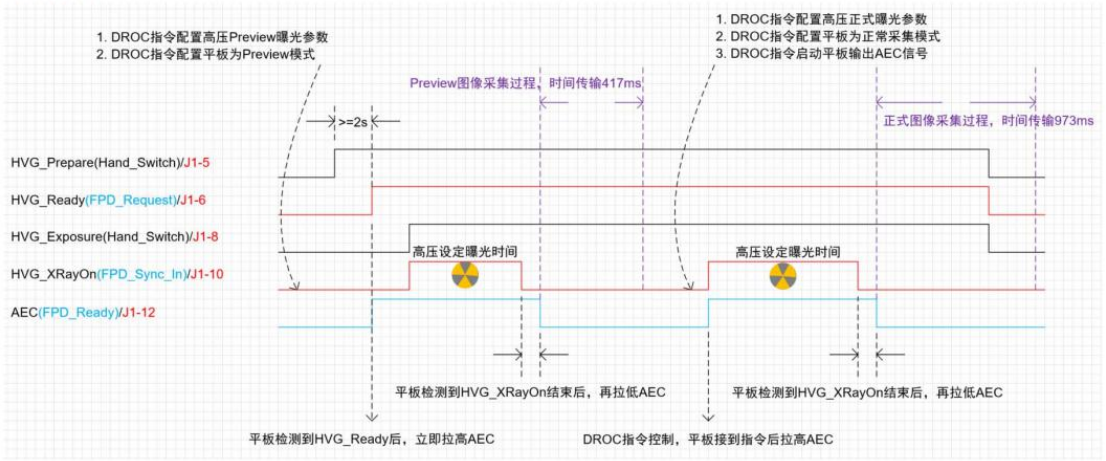
4、下次采集为正常图像。

最后，注意图像分辨率和数据缓冲区的大小随 Preview 模式设置而改变。

10、SAEC 模式

S-AEC曝光

S-AEC 曝光模式下，曝光分为预曝光和正式曝光两个部分，第一帧进行预曝光，系统根据预曝光图像调节第二帧正式曝光的参数。在曝光期间，高压支持调节曝光参数，每个曝光流程的正常终止时间是根据设定的 ms 或者 mAs 相关，整个曝光序列的时间不能超过 30s，包括 Prep 的准备 2s 时间。



HVG	FPD	Connector	Function
HVG_Prepare		J1-5	Handswitch 1st gear
HVG_Ready	FPD_Request	J1-6	Flat panel detector request in
HVG_Exposure		J1-8	Handswitch 2nd gear
HVG_XRayOn	FPD_Sync_In	J1-10	Flat panel detector AEC turn off
AEC	FPD_Ready	J1-12	Flat panel detector AEC control out

注意：

1、SAEC 模式仅支持 Panel-2530-85um 平板；

2、SAEC 模式的触发模式为:trigger mode=0x08;

3、步骤：

1》调用 HBI_SAecModeApply 接口函数，其中 preview 为 1 表示为上传 preview 图像，反之为 0 上传正常图像，

m_nPerReadyTm, m_nPostReadyTm 分别为 FPD per ready delay time 和 FPD post ready delay time；

2》调用 HBI_SAecModeApply 接口函数，其中 preview 为 0 表示为上传正常图像，另外 m_nPerReadyTm 和 m_nPostReadyTm 非关键参数默认即可；

3》调用 HBI_SAecAcq 接口函数完成正常采图；

4》回调函数中事件 ID:

CASE ECALLBACK_TYPE_SINGLE_IMAGE:正常图像,例如 Panel-2530-85um，分辨率默认为 width=2816，height=3584；

CASE ECALLBACK_TYPE_PREVIEW_IMAGE:previe 图像，例如 Panel-2530-85um，分辨率默认为:wwidth=2816/4=704,height=3584/4=896;

```
void CSAecModeTester::OnBnClickedBtnSaecAutoTest()
{
	// TODO: 在此添加控件通知处理程序代码
	this->UpdateData(TRUE);
	//
	if (!IsOpen) {
		AfxMessageBox(_T("err:FPD is disconnect!"));
		return;
	}
	//
	if (m_pRegRomCfg == NULL) {
		AfxMessageBox(_T("err:m_pRegRomCfg!"));
		return;
	}
	//
	if (m_pRegRomCfg->m_SysBaseInfo.m_byPanelSize != 0x05) {
		AfxMessageBox(_T("err:FPD is not Pane2530-85um!"));
		return;
	}
	//
	if (m_nPerReadyTm * m_nPostReadyTm <= 0) {
		AfxMessageBox(_T("err:Ready Time is error!"));
		return;
	}
	// STEP1
	theDoc->ReportError(_T("STEP1:SAEC Preview-Mode Start!"));
	int preview = 0x01; // 01-Preview Image;00-Normal Image;
	int ret = HBI_SAecModeApply(m_pFpd, preview, m_nPerReadyTm, m_nPostReadyTm);
	if (ret != HBI_SUCCSS) {
```

```
        AfxMessageBox(_T("err:Set SAEC Preview-Mode failed!"));
        return;
    }
    ReportError(_T("STEP1:SAEC Preview-Mode Finished!"));

    // 等待上传 preview 图像
    ReportError(_T("STEP1:SAEC Preview-Image!"));
    // 。。。。。。
    ReportError(_T("STEP1:SAEC update hvg dose!"));
    // 分析 preview 图像更改剂量

    // STEP2
    ReportError(_T("STEP2:SAEC Normal-Mode Start!"));
    preview = 0x00; // 01-Preview Image;00-Normal Image;
    ret = HBI_SAecModeApply(m_pFpd, preview, m_nPerReadyTm, m_nPostReadyTm);
    if (ret != HBI_SUCCSS) {
        AfxMessageBox(_T("err:Set SAEC Normal-Mode failed!"));
        return;
    }
    ReportError(_T("STEP2:SAEC Normal-Mode Finished!"));
    // STEP3
    ReportError(_T("STEP3:SAEC Acquition Image Start!"));
    ret = HBI_SAecAcq(m_pFpd);
    if (ret != HBI_SUCCSS) {
        AfxMessageBox(_T("err:Send SAEC Acquition Image!"));
        return;
    }
    ReportError(_T("STEP3:SAEC Acquition Image Finished!"));
}
```

11、开发 DMEO

为了便于用户快速开发集成特提供开发 Demo 源码，见 SDK_Demo_Example 目录

DemoLib 为静态开发示例，使用 C/C++语言，基于 vs2017 平台+MFC 开发，功能比较齐全。

DemoDll 为动态开发示例，使用 C/C++语言，基于 vs2017 平台+control 开发，实现基本功能。

HBI_FPDCSharpdem 使用 C#语言，基于.net 平台开发，功能比较齐全。