昊博 SDK 双板开发 Demo 工程说明

昊博影像研发部

更新记录

版本号	日期	描述	作者
Ver 4.0.0.7	2021.5.25	文档创建	MH.YANG
Ver 4.0.0.13	2021.10.11	定版	MH.YANG
Ver 4.0.0.15	2021.11.19	双板	MH.YANG
Ver 4.0.1.6	2022.07.19	双板:路由器或交换机方案	MH.YANG

目录

目的：针对二次开发集成用户，易于掌握、方便调试和快速集成。

一、开发环境

win7 以上+vs2015+MFC+Opencv341

设置安装 PCIe 卡驱动，详细见附件 SDK API 接口开发说明书。

显示部分基于 Opencv341 开发，请先安装 Opencv341。

驱动安装

基于光口 PCIe 卡的平板，请参考《昊博 PCIe 设备驱动安装说明_***.pdf》，基于网口的平板网络配置，部分平板需要设置网口 Jumbo 选项，具体请参考《昊博 FPD Pro SDK API Programming Reference Ver*.*.*.*.pdf》文档。

依赖库见百度网盘

链接：https://pan.baidu.com/s/122AEkzb3a_otibASg7SIYA

提取码：7az0

二、功能说明

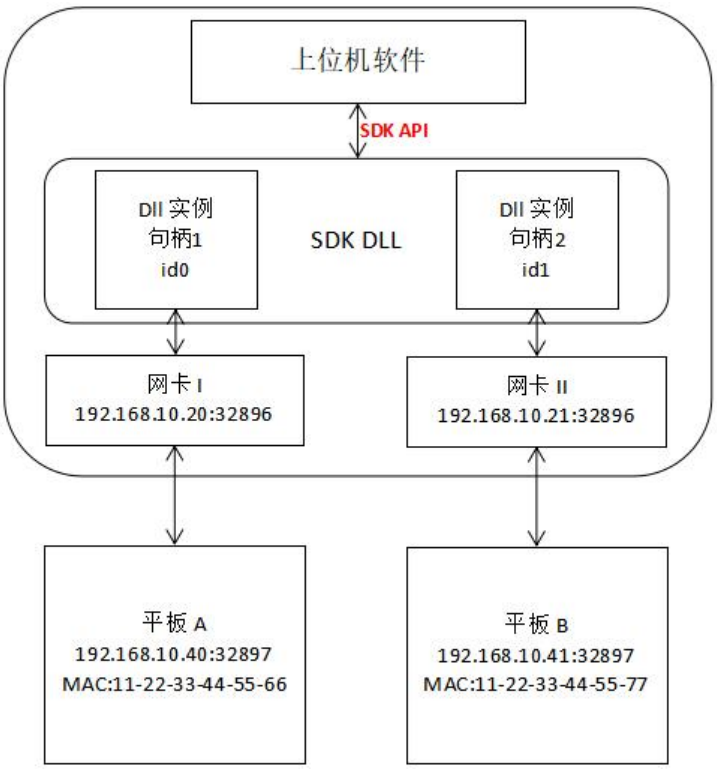
该版本 SDK 支持静态网口、动态网口以及 PCIe 光口、无线方式通讯，支持不同的通讯方式平板组成的 “双板”解决方案，比如双千兆网卡或千兆网卡与 PCIe 卡组成的双平板解决方案。

2.1、“双板”环境

下面主要介绍基于网络通讯方式组成的“双板”方案，常见两种方式：双网卡方式和路由器或交换机方式。

2.1.2、双网卡方式

“双网卡方式”即 PC/工控机端安装有两个千兆网卡，两个平板探测器分别直连到两个网卡上，如下图：



1、两个网卡必须都支持千兆网络的带宽。

2、只有网卡支持巨帧设置才可以支持巨帧版本的平板探测器。

3、平板 A 和平板 B 必须拥有不同的 MAC 地址和 IP 地址；

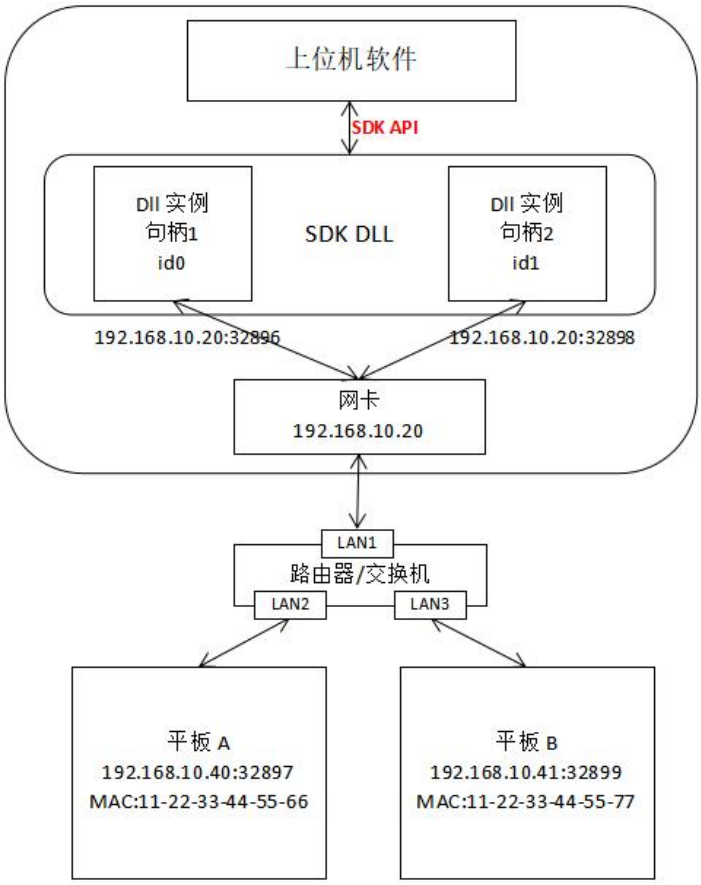
4、网卡 A 与平板 A 通讯，网卡 B 与平板 B 通讯，网卡 A 和平板 A 同一网段，网卡 B 和平板 B 同一网段，网卡 A 和网卡 B 地址不一样，可以同一网段也可以在不同网段；

5、平板默认 IP 地址和 MAC 地址 192.168.10.40 和 11-22-33-44-55-66，需要修改平板 B 的 IP 地址和 MAC 地址，具体请参考 XDiscovery 软件操作使用说明书。

2.1.2、路由器或交换机方式

该方式通过中间设备（路由器或交换机）组成的局域网，实现 PC/工控机端通过一个网卡同时控制局域网中的两台平板探测器。本地网卡、路由器以及交换机支持千兆网。

如下图：



1、本地网卡、路由器以及交换机支持千兆网络的带宽；

2、网卡、路由器以及交换机支持巨帧设置才可以支持巨帧版本的平板探测器。

3、平板 A 和平板 B 必须拥有不同的 MAC 地址和和同一网段的不同的 IP 地址、端口；

4、本地网卡通过**端口 1（32896）**与平板 A 通讯，网卡通过**端口 2（32898）**与平板 B 通讯，前提是本地网卡和网平板 A 和平板 B 为同一网段；

5、平板默认 IP 地址和 MAC 地址 192.168.10.40:32897 和 11-22-33-44-55-66 需要修改平板 B 的 IP 地址和 MAC 地址，如（192.168.10.41:32899 和 11-22-33-44-55-77），具体请参考 XDiscovery 软件操作使用说明书。

双板”解决方案较于“单板”方案，上位机注意区分控制不同的平板的参数和图像数据。

其他和“单板”雷同，见 具体参考 API 说明书和 SDK Demo 源码。

2.2、主界面

如下图 1-1 所示，分为 4 个主要区域：**菜单栏、图像显示窗口、连接/配置/采图/下载/控制窗口、图像显示/保存以及配置查看按钮。**

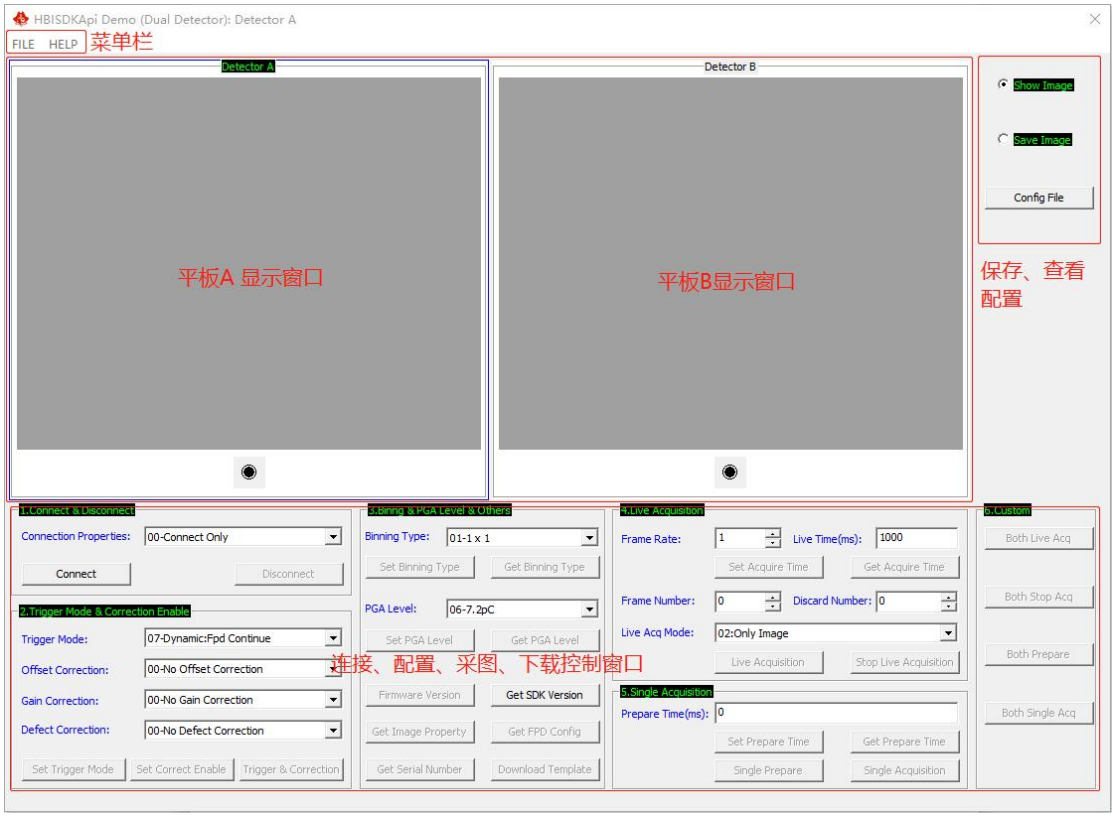


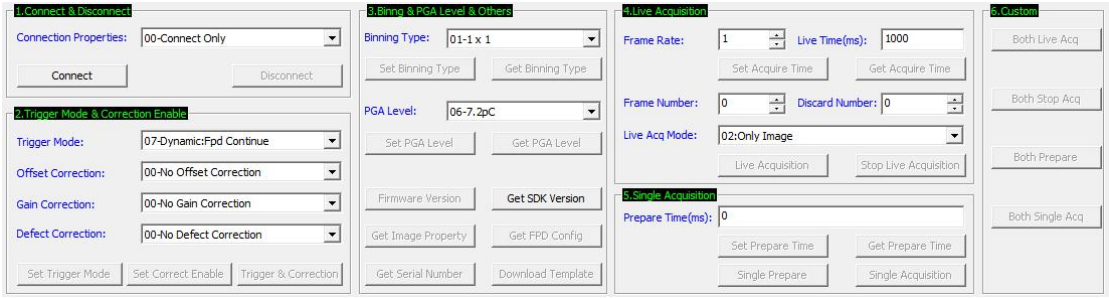
图 1-1

菜单栏-**子菜单“Detector Setting”-》弹出平板通讯配置框，用户可以通过该窗口设置双板的通讯参数；**

菜单栏-**子菜单“快速生成模板”，用户可以通过该窗口生成校正模板；**

图像显示窗口-可以同时显示平板 A 和平板 B 的图像。目前没有进行自适应窗宽窗位算法，仅为调试；

状态、控制参数窗口-**平板状态（连接、断开、ready、busy、offset、gain 或 defect 等）、平板地址端口、触发模式、校正使能选项、帧率设置、PGA 档位调节、Binning 模式类型、采集方式（生成模板并显示图像、仅显示图像和仅生成模板）等；**



主要模块如下表：

模块	描述
1、Connect & Disconnect	连接平板和断开连接
2、Trigger Mode & Correction Enable	设置触发模式和矫正使能
3、Binning & PGA & Others	设置/获取 Binning 类型、PGA 档位，获取 SDK、固件版本号，获取图像属性，下载模板
4、Live Acquistion	连续采集操作，包括参数连续采集控制参数
5、Single Shot Acquistion	单帧采集操作，包括参数单帧采集控制参数
6、Custom	同时连续采集、单帧采集、停止采集等

图像显示/保存以及配置查看按钮：当选择“show Image”显示图像，“Save Image”保存图像。配置查看主要查看配置记录，为 ini 文件。

当前操作只针对“主”探测器。

2.3、调试信息打印窗口

如下图 1-2 所示，打印当前调试信息等。

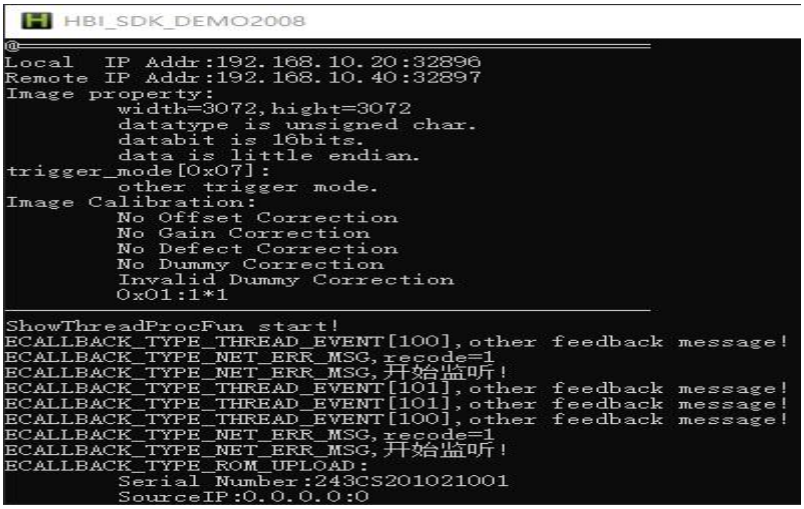
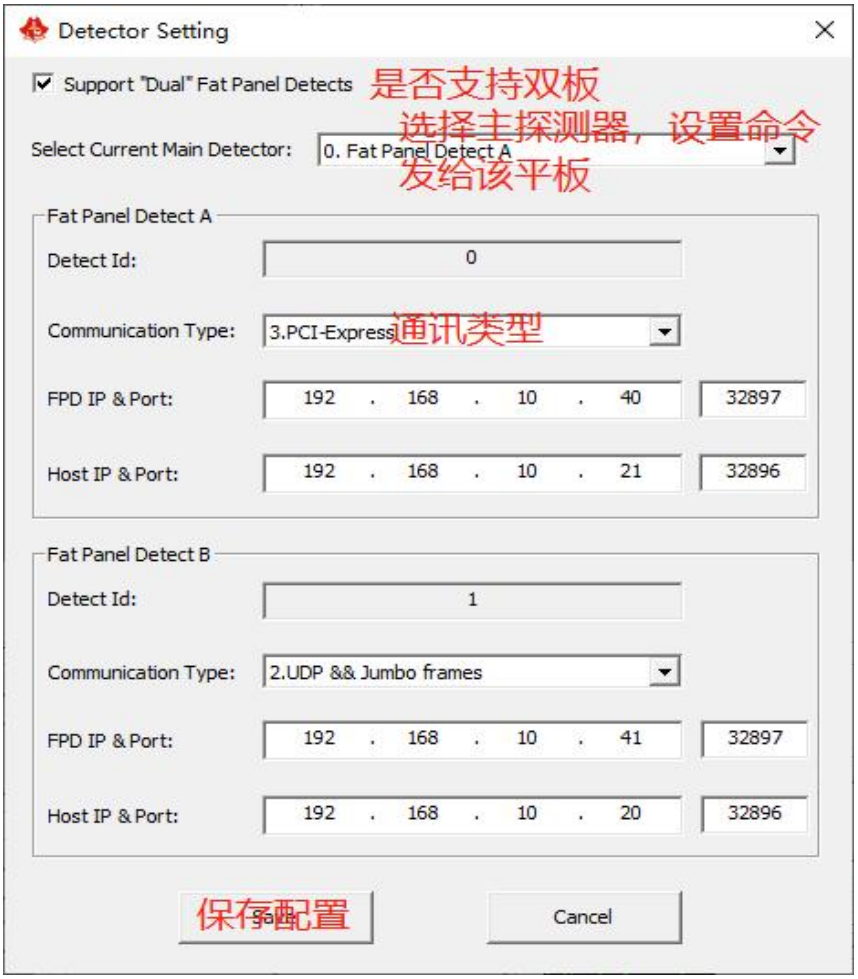


图 1-2

2.4、平板配置界面

菜单栏-子菜单**“Detector Setting”-》弹出平板通讯配置框**，用户在这里设置双板的通讯参数；



点击 Config File，查看配置文件《hbiapidll.ini》。

```
[HBI_CFG]                    #  通讯配置
IS_DUAL=1                    # 支持双板 1 还是单板 0
DEFAULT_ID=0                 # 默认主板，0-平板 A,1-平板 B
[DETECTOR_A]                 # 平板 A
DETECTOR_ID=0                # 平板 Aid
COMM_TYPE=2                  # 通讯方式，0-udp，1-udp jumbo，2-pcie 光口
DETECTOR_IP=192.168.10.40    # 平板 A ip
DETECTOR_PORT=32897          # 平板 A 端口
LOCAL_IP=192.168.10.20       # 本地网卡 1 ip
LOCAL_PORT=32896             # 本地网卡 1 端口 port
[DETECTOR_B]                 # 平板 B
DETECTOR_ID=1                # 平板 B id
COMM_TYPE=1                  # 通讯方式，0-udp，1-udp jumbo，2-pcie 光口
DETECTOR_IP=192.168.10.40    # 平板 B ip
DETECTOR_PORT=32897          # 平板 B 端口
LOCAL_IP=192.168.10.20       # 本地网卡 2 ip
LOCAL_PORT=32896             # 本地网卡 2 端口 port
```

2.5、快速生成模板界面

菜单“File”-》子菜单“Generate Template”-》弹出对话框如下图 1-3 所示，生成 Offset、Gain 和 Defect 模板。

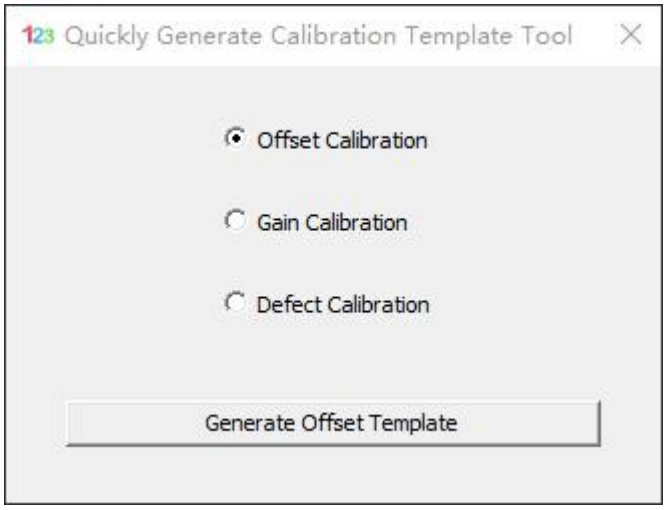


图 1-3

✓ Offset 模板

需要采集暗场图像，只需要采集一组图像；

步骤：

Offset 模板一般需要每天做一次，有两种情况说明：

1》不存在 offset 模板或每天开机后先完成 offset 模板的制作。

2》中途有平板断电状况等，再次连接需要下载 offset 模板到固件，否则出现失校正现象。

3》客户可根据 offset 模板文件判断是否重做 offset 模板或者下载模板。

例如：offset 模板不存在或者文件属性创建日期为 T-1 的，重做 offset 模板，上位机发送做 offset 模板命令，固件完成 offset 模板并在本地保存一张 offset 模板；如果 offset 模板为当天模板，下载 offset 模板到固件即可。

✓ Gain 模板

需要采集亮场图像，需要清场，不能在平板上放置物品等，

采集 1 组亮场场图，整常高压，毫安秒调节正常的 50%。

步骤：

1》设置固件 Offset 使能：03-preoffset correction；

2》调节好剂量，采集一组亮场并成成模板；

3》将 gain 模板下载到固件；

✓ Defect 需要采集暗场图像，只需要采集一组图像

defect 模板需要采集 3 组亮场图像，需要清场，不能在平板上放置物品等

Group1：正常高压，毫安秒调节正常的 10%

Group2：正常高压，毫安秒调节正常的 50%

Group3：正常高压，毫安秒调节正常

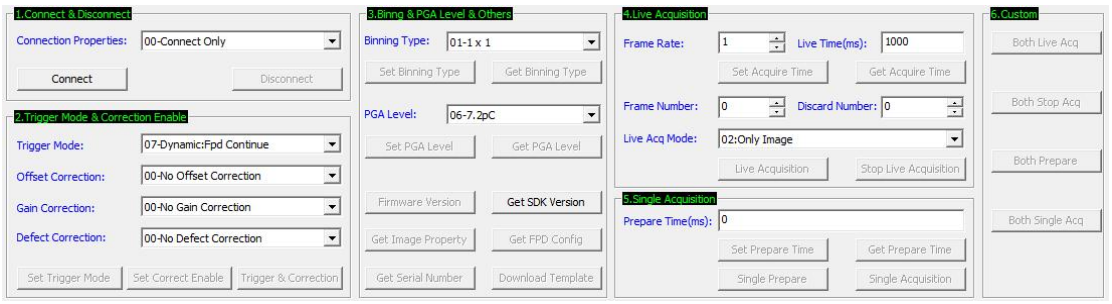
步骤：

1》设置固件 Offset 使能：03-preoffset correction；

2》调节剂量采集第一组亮场，以此采集完 3 组亮场，注意不同剂量调节，后将自动生成 defect 模板；

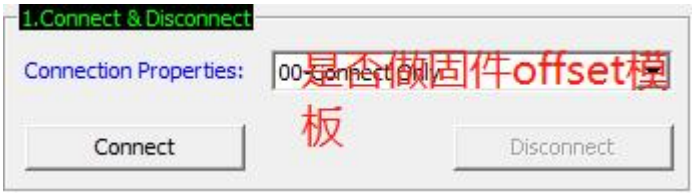
3》下载 defect 模板到固件；

2.6、功能界面



连接、配置、采图、下载、矫正使能、切换、获取版本信息。

1》连接、断开



前提：使用前先初始化句柄和注册回调函数：HBI_Init、HBI_RegEventCallBackFun，初始化后将获得句柄指针，对该句柄注册回调函数，回调函数反馈消息和数据。

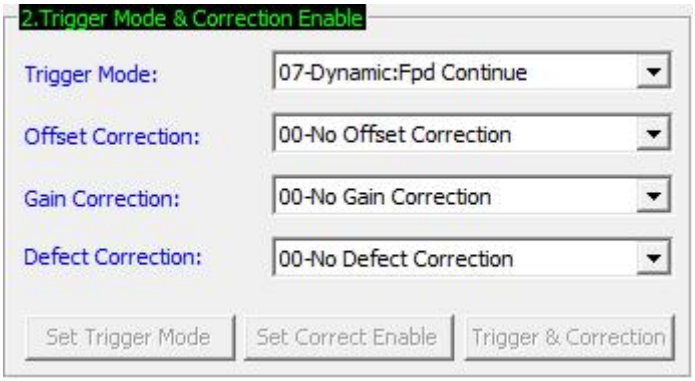
接口函数：HBI_ConnectDetector 和 HBI_DisConnectDetector。

连接函数参数包括 3 部分：**句柄、通讯参数和是否做固件 offset 模板。**

连接平板，如果最后一个参数为 1，连接成功后将自动做固件 offset 模板，非 1 或默认不做固件 offset 模板，如果做 offset 模板确认球管是关闭状态。

COMM_CFG commCfg 是通讯参数，包括以太网 UDP、UDP Jumbo 和 PCIe 光口方式，其中以太网方式需要设置正确的 IP 地址和端口号。

2》设置触发模式和矫正使能

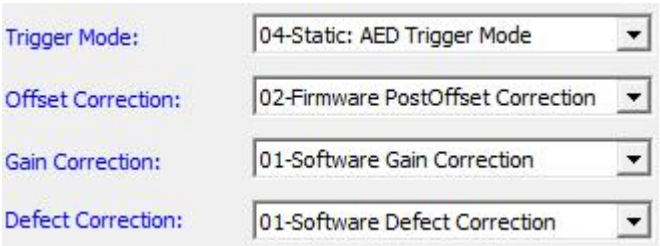


✓ **触发模式**

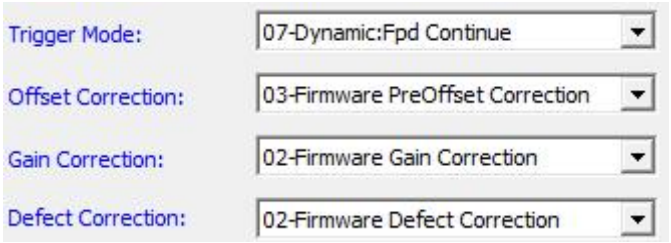
静态平板，即通讯方式为 UDP 方式的低帧率平板，常用的触发模式为 01-software trigger mode；03-HVG trigger mode；04-Free AED trigger mode；
动态平板，即通讯方式为 UDP Jumbo 或 PCIe 光口方式的高帧率平板，，常用的触发模式为 07-Dynamic：Fpd Continue mode；

✓ **矫正使能**

静态平板采用软件矫正，即 SDK 采完图矫正图像。
软件校正使能的必要条件：制作完矫正模板，使能后自动加载默认目录下模板文件。
目前矫正方式如下图



动态平板一般使用固件校正，即校正在平板中完成，SDK 只完成图像传输和模板制作以及下载工作。
固件校正使能的必要条件：制作完模板和已经下载到固件。
Offst、Gain 和 defect 逐次依赖，即 gain 校正需要先做 offset 校正，defect 校正需要先做 offset 和 gain 校正，否则效果不好。
默认校正使能如下图所示。



✓ **接口函数**

HBI_UpdateTriggerMode：设置触发模式；
HBI_UpdateCorrectEnable：设置矫正使能状态；
HBI_TriggerAndCorrectApplay：设置触发模式和矫正使能状态；

参数和参数类型参考头文件或 API 接口文档以及 SDK Demo 工程源码。

3》设置/获取 Binning、PGA 档位，获取 SDK、固件版本号、序列号、平板参数、下载模板

接口函数

HBI_SetBinning/HBI_GetBinning：设置/获取 binning 类型；

HBI_SetPgaLevel/HBI_GetPgaLevel：设置/获取 PGA 档位；

HBI_GetFirmareVerion:获取固件版本号；

HBI_GetSDKVerion:获取 sdk 版本号；

HBI_GetFPDSerialNumber:获取平板序列号；

HBI_GetImageProperty:图像属性；

HBI_GetFpdCfgInfo:获取平板参数；

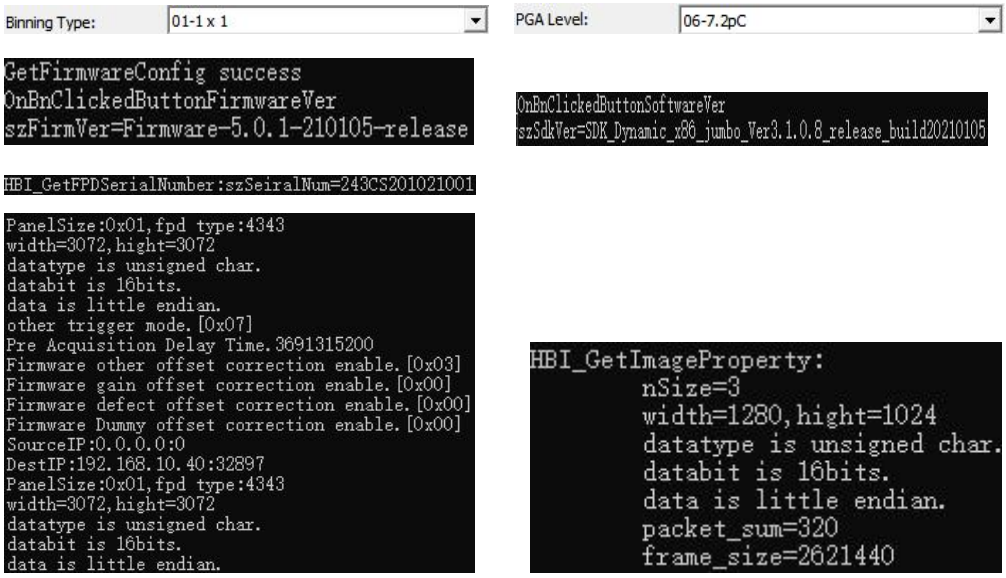
下载模板：

首先注册下载进度回调函数：HBI_RegProgressCallBack。

可选择下载函数 1 或下载函数 2

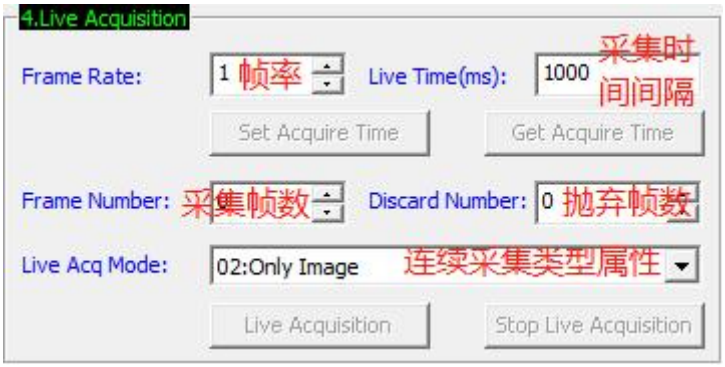
下载函数 1：HBI_DownloadTemplate

下载函数 2：HBI_DownloadTemplateByType



参数和参数类型参考头文件或 API 接口文档以及 SDK Demo 工程源码。

4》连续采集



接口函数：

HBI_SetLiveAcquisitionTime:设置连续采集时间间隔，即与帧率匹配，如 1fps 的时间间隔是 1000ms，2fps 的时间间隔是 500ms；

HBI_GetLiveAcquisitionTime:获取连续采集时间间隔；

HBI_LiveAcquisition：连续采集命令；

HBI_StopAcquisition：停止连续采集命令；

连续采集参数：

采集命令：LIVE_ACQ_DEFAULT_TYPE

Frame Number:0 表示一直采集，知道发送停止采集命令；n>=1,采集 1~n 张图像。

Discard：表示抛弃前 n 张图像。

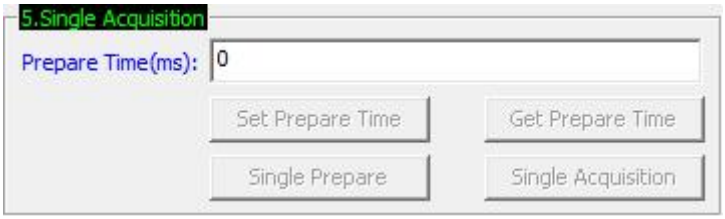
Live Acq Mode：表示连续采集模式：

01-Image+Offset Template：表示做完 Offset 模板开始采图；

02-Only Image：表示只采图；

03-Only Offset Template：表示只做 Offset 模板；

5》单帧采集



设置 Prepare 延时：

HBI_SetSinglePrepareTime:设置单帧 Prepare 延时，0 表示或大于 0；

HBI_GetSinglePrepareTime:获取单帧 Prepare 延时；

单帧采集有两种模式：

模式一：Prepare 延时大于 0 时，触发 HBI_SinglePrepare 命令，固件收到命令自清空并延时后自动上传一帧图像；

模式二：Prepare 延时等于 0 时，采集一帧的流程是触发 HBI_SinglePrepare 命令-》固件收到命令自清空-》用户延时-》触发 HBI_SingleAcquisition 命令-》上传一帧图像；

单帧采集参数：使用 HBI_SingleAcquisition 接口参数，采集命令：SINGLE_ACQ_DEFAULT_TYPE，其他参数默认

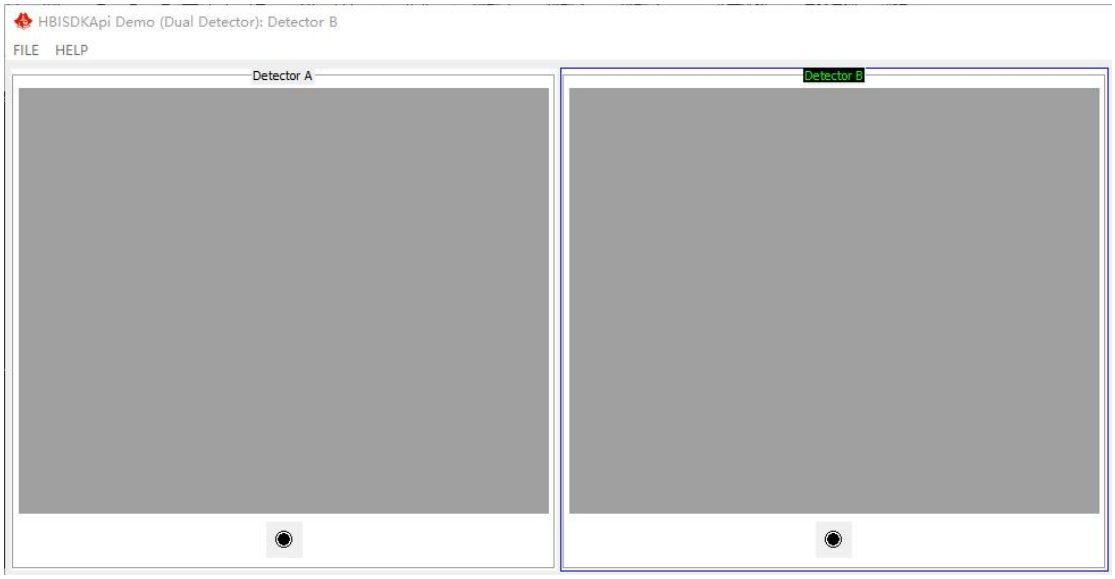
6》双板同时采集

如果同时采集或停止采集，可以通过句柄向平板分辨发送采集命令即可，与上面单帧和连续采集类似，具体参考 SDK Demo 工程源码，这里不再赘述。

2.7、平板切换

点击显示窗口，完成探测器 A 和探测器 B，点击探测器 B 窗口，探测器 B 切换为“主”探测器，如下图：

点击探测器 B 窗口，探测器 B 切换为“主”探测器。



2.8、显示图像和保存图像

针对当前“主”平板采图显示和保存切换。

三、流程说明

以本工程为例说明

HBI_DLL 为 SDK 对外接口文件。

HBI_DLL\INCLUDES：包含 HbiFpd.h、HbiType.h 和 HbiError.h 一共三个头文件。

a.《HbiFpd.h》：导出函数以及说明，具体可参考《昊博 FPD PCIe SDK API Programming Reference Verr*.*.*.*.pdf》；

b.《HbiType.h》：命令、回调函数定义返回事件类型以及固件参数结构体；

c.《HbiError.h》：错误以及返回码信息表；

HBI_DLL\BIN：动态库库文件，注意 32bits 和 64bits；

HBI_DLL\DOC：API 接口函数说明文档。

主流程：

1》连接成功后，平板会自动反馈 ROM 参数。

2》HBI_Init 和 HBI_Destroy：初始化和释放设备对应；

3》HBI_ConnectDetector 和 HBI_DisConnectDetector：连接和断开平板，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

4》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。

5》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。

6》HBI_GetSysParamCfg 和 HBI_GetFpdCfgInfo：都是读取平板固件全部参数，HBI_GetSysParamCfg 是向固件

发请求获取全部参数，异步函数；HBI_GetFpdCfgInfo 是连接成功或设置成功后获取全部参数，同步函数。

3.1、头文件和库文件导入

下面主要介绍“双板”的集成流程。

1》添加头文件

将"HBI_DLL\INCLUDES"头文件拷贝的用户开发指定目录下；

将动态库文件"HBI_DLL\BIN"拷贝到用户工程目录或者系统目录下，注意 32bits 和 64bits；

2》接口调用流程

```
#define _DLL_EX_IM 0
#include "HbiFpd.h"
#pragma comment(lib,"HBISDKApi.lib")
```

3》定义基本数据类型和函数方法

```
void* m_pFpd;                // 当前 DLL 实例句柄
bool m_IsOpen;               // 该平板是否处于连接状态
CFPD_BASE_CFG *m_fpd_base; // 当前探测器基础数据
RegCfgInfo* m_pLastRegCfg; // 固件配置
FPD_AQC_MODE m_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
IMAGE_PROPERTY m_imgApro, m_imgBpro;// 平板 A 和平板 B 的图像属性

// 回调函数以及显示线程
// 回调函数可以定义一个，根据平板 ID 切换，当前 Demo 为了支持同时采集显示
static int handleCommandEventA(void* _contex, int nDevId, unsigned char byteEventId, void
*buff, int param2, int param3, int param4);
static int handleCommandEventB(void* _contex, int nDevId, unsigned char byteEventId, void
*buff, int param2, int param3, int param4);
static unsigned int __stdcall ShowThreadProcFunA(LPVOID pParam);
static unsigned int __stdcall ShowThreadProcFunB(LPVOID pParam);

// 用户根据实际自定义
int ShowImageA(unsigned short *imgbuff, int nbufflen, int nframeid, int nfps);
int ShowImageB(unsigned short *imgbuff, int nbufflen, int nframeid, int nfps);
int SaveImage(unsigned char *imgbuff, int nbufflen, int type = 0);
void UpdateImageA();
void UpdateImageB();

// 省略 ... ...
```

3.2、初始化 DLL 资源

```
void *handle1 = pBaseCfg1->m_pFpdHand;
void *handle2 = pBaseCfg2->m_pFpdHand;
#if 1
if (handle1 == NULL && handle2 == NULL)
```

```
{
    HBI_HANDLE handleArr[DETECTOR_MAX_NUMBER];
    int nArrSize = sizeof(handleArr) / sizeof(HBI_HANDLE);
    int ret = HBI_InitEx(handleArr, nArrSize);
    if (ret != HBI_SUCCSS)
    {
        ::AfxMessageBox(_T("err:HBI_InitEx failed!"));
        return;
    }
    //
    handle1 = handleArr[0]._handle;
    handle2 = handleArr[1]._handle;
}
#else
if (handle1 == NULL && handle2 == NULL)
{
    int ret = HBI_InitDual(&handle1, &handle2)
    if (ret != HBI_SUCCSS)
    {
        ::AfxMessageBox(_T("err:HBI_InitDualfailed!"));
        return;
    }
}
#endif
```

3.3、设置注册回调函数

```
// Detector A 注册回调函数
int ret = HBI_RegEventCallBackFun(handle1, handleCommandEventA);
if (ret != 0)
{
    printf("Detector A:HBI_RegEventCallBackFun failed!\n");
    HBI_Destroy(handle1);
     handle1 = NULL;
    return false;
}
// Detector B 注册回调函数
ret = HBI_RegEventCallBackFun(handle2, handleCommandEventB);
if (ret != 0)
{
    printf("Detector B:HBI_RegEventCallBackFun failed!\n");
    HBI_Destroy(handle2);
     Handle2 = NULL;
    return false;
}
```

3.4、连接平板

```
COMM_CFG commCfg;
//doOffsetTemp - 非1:连接成功后固件不做offset模板，1:连接成功后固件做offset模板
int nOffsetTemp = 0; //如果默认即固件不做offset模板
// Detector A连接
//Detector A通讯方式：UDP类型
commCfg._type = FPD_COMM_TYPE::UDP_COMM_TYPE;
strcpy(commCfg._localip,  “192.168.10.20”);
strcpy(commCfg._remoteip, “192.168.10.40”);
commCfg._loacalPort = 32896;
commCfg._remotePort = 32897;
ret = HBI_ConnectDetector(handle1, commCfg, nOffsetTemp);
if (ret != 0)
{
    printf("连接失败!\n");
    return false;
}
// Detector B连接
// Detector B通讯方式：UDP Jumbo类型
commCfg._type = FPD_COMM_TYPE::UDP_JUMBO_COMM_TYPE;
strcpy(commCfg._localip,  “192.168.10.21”);
strcpy(commCfg._remoteip, “192.168.10.41”);
commCfg._loacalPort = 32896;
commCfg._remotePort = 32897;
ret = HBI_ConnectDetector(handle2, commCfg, nOffsetTemp);
if (ret != 0)
{
    printf("连接失败!\n");
    return false;
}
```

3.5、设置触发模式和图像矫正使能

```
int _triggerMode = 1; // 1-软触发，3-高压触发，4-freeAED
IMAGE_CORRECT_ENABLE* pCorrect = new IMAGE_CORRECT_ENABLE;
if (pCorrect == NULL) return 0;

// Detector A设置触发模式和矫正使能
pcorrect->bFeedbackCfg = true;       //true-ECALLBACK_TYPE_ROM_UPLOAD
Event,false-ECALLBACK_TYPE_SET_CFG_OK Event
pcorrect->ucOffsetCorrection = 0x02; //0-No Offset Correction;1-Software PreOffset
Correction;2-Firmware PostOffset Correction;3-Firmware PreOffset Correction;
pcorrect->ucGainCorrection   = 0x01; //0-"Do nothing";1-"Software Gain Correction";
2-"Hardware Gain Correction"
```

```
pcorrect->ucDefectCorrection = 0x01; //0-"Do nothing";1-"Software Defect Correction";
2-"Software Defect Correction"
pcorrect->ucDummyCorrection  = 0x00; //0-"Do nothing";1-"Software Dummy Correction";
2-"Software Dummy Correction"
ret=HBI_TriggerAndCorrectApplay(handle1,_triggerMode, pCorrect);
if (ret != 0)
{
    printf("连接失败!\n");
}

// Detector B 设置触发模式和矫正使能
_triggerMode = 7;
pcorrect->bFeedbackCfg = true;       //true-ECALLBACK_TYPE_ROM_UPLOAD
Event,false-ECALLBACK_TYPE_SET_CFG_OK Event
pcorrect->ucOffsetCorrection = 0x03; //00-"Do nothing";01-"prepare Offset Correction";
02-"post Offset Correction";
pcorrect->ucGainCorrection   = 0x02; //00-"Do nothing";01-"Software Gain Correction";
02-"Hardware Gain Correction"
pcorrect->ucDefectCorrection = 0x02; //00-"Do nothing";01-"Software Defect Correction";
02-"Software Defect Correction"
pcorrect->ucDummyCorrection  = 0x00; //00-"Do nothing";01-"Software Dummy Correction";
02-"Software Dummy Correction"
ret=HBI_TriggerAndCorrectApplay(handle2,_triggerMode, pCorrect);
if (ret != 0)
{
    printf("连接失败!\n");
}
```

3.6、断开连接或释放资源

```
//HBI_DisConnectDetector(handle1);
//HBI_DisConnectDetector(handle2);
// 回收资源(包括断开连接和资源释放)
//HBI_Destroy(handle1);
//HBI_Destroy(handle2);
HBI_DestroyEx();
handle1 = NULL;
Handle2 = NULL;
```

3.7、注册回调函数

```
static int handleCommandEvent(void* _contex, int nfpdId, unsigned char command, void *pvParam1, int nParam2,
int nParam3, int nParam4);
class CXXX {
    ...
public:
```

static int handleCommandEvent(void* _contex, unsigned char command, void *buff, int len, int nid, int param2, int param3);

...

};

参数说明：

// @USER_CALLBACK_HANDLE_ENVENT

// @pContext:参数 1，上位机对象指针，可以为空（NULL）

// @ ufpdId:参数 2，例如平板 id

// @byteEventid:参数 3，事件 ID，参考 HbiType.h 中 enum eCallbackEventCommType

// @PVEventParam1:参数 4，配置指针或图像结构体指针

// @nEventParam2:参数 5，例如 data size 或状态

// @nEventParam3:参数 6，例如帧号 frame id

// @nEventParam4:参数 7，例如帧率 frame rate 或状态等

回调函数返回：

事件 ID	值	备注
ECALLBACK_TYPE_FPD_STATUS	0X09	平板状态
ECALLBACK_TYPE_ROM_UPLOAD	0X10	平板 ROM 参数
ECALLBACK_TYPE_SET_CFG_OK	0XAB	设置参数成功（一般返回 ROM 参数）亦可返回成功消息。
ECALLBACK_TYPE_SINGLE_IMAGE	0X51	单帧采集图像
ECALLBACK_TYPE_MULTIPLE_IMAGE	0X52	连续采集图像
ECALLBACK_TYPE_OFFSET_TMP	0X5C	固件做完 pre-offset 上传 offset 模板数据
ECALLBACK_TYPE_PREVIEW_IMAGE	0X53	Preview 模式下上传 preview 图像
ECALLBACK_OVERLAY_NUMBER	0X5D	图像叠加，反馈叠加过程，即叠加到第几帧
ECALLBACK_OVERLAY_16BIT_IMAGE	0X5E	叠加结果为 16 位图像数据
ECALLBACK_OVERLAY_32BIT_IMAGE	0X5F	叠加结果为 32 位图像数据
ECALLBACK_TYPE_PACKET_MISS	0X5B	丢包/丢帧反馈信息
ECALLBACK_TYPE_LIVE_ACQ_OK	0XA0	反馈保存图像文件的信息:包括目录、平均灰度值
ECALLBACK_TYPE_THREAD_EVENT	0XA5	监控线程或应用线程监控信息
ECALLBACK_TYPE_PACKET_MISS_MSG	0XA4	采集图像中发现丢包并反馈丢包信息
ECALLBACK_TYPE_ACQ_DISCARDDED	0XA6	连续采集中抛帧，sdk 反馈抛帧记录
ECALLBACK_TYPE_OFFSET_ERR_MSG	0XA7	分布生成 offset 模板结果
ECALLBACK_TYPE_GAIN_ERR_MSG	0XA8	分布生成 gain 模板结果
ECALLBACK_TYPE_DEFECT_ERR_MSG	0XA9	分布生成 defect 模板结果
ECALLBACK_TYPE_UPDATE_FIRMWARE	0X06	固件更新：进度以及结果
ECALLBACK_TYPE_ERASE_FIRMWARE	0X07	固件更新：擦除固件
ECALLBACK_TYPE_WORK_STATUS	0XAF	工作模式：生产模式和调试模式切换反馈
ECALLBACK_TYPE_SAVE_SUCCESS	0XAC	常见生成模板过程中保存图像反馈
ECALLBACK_TYPE_GENERATE_TEMPLATE	0XAD	快速生成模板信息反馈，注意与分步生成区别
ECALLBACK_TYPE_FILE_NOTEXIST	0XAE	查找文件不存在反馈

例：平板状态(ECALLBACK_TYPE_FPD_STATUS)

状态	值	备注
FPD_DISCONN_STATUS	0	回调收到-11《 nEventParam2 《0 都视为通讯异常，详细参考 SDK Demo 工程
FPD_PREPARE_STATUS	1	单帧采集，清空
FPD_READY_STATUS	2	静态平板 Ready 状态，平板空闲状态，可以采图
FPD_DOOFFSET_TEMPLATE	3	固件生成 pre-offset 模板中
FPD_EXPOSE_STATUS	4	采图中
FPD_CONTINUE_READY	5	动态平板 Ready 状态，平板空闲状态，可以采图
FPD_DWONLOAD_GAIN	6	下载 Gain 模板中
FPD_DWONLOAD_DEFECT	7	下载 Defect 模板中
FPD_DWONLOAD_OFFSET	8	下载 Offset 模板中
FPD_UPDATE_FIRMARE	9	更新固件中
FPD_RETRANS_MISS	10	重传数据中
FPD_STATUS_AED	11	AED 采图中
FPD_STATUS_SLEEP	12	无线平板低功耗：休眠
FPD_STATUS_WAKEUP	13	无线平板低功耗：唤醒
FPD_DOWNLOAD_NO_IMAGE	14	无线平板下载图像：无图像可下载
FPD_DOWNLOAD_TAIL_IMAGE	15	无线平板下载图像：下载最后一帧图像
FPD_EMMC_MAX_NUMBER	16	Emmc 存储容量报警
FPD_ENDTIME_WARNNING	17	固件授权结束
FPD_CONN_SUCCESS	100	连接成功

回调函数返回：

case ECALLBACK_TYPE_NET_ERR_MSG: // 平板状态
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

```
case ECALLBACK_TYPE_FPD_STATUS: // 平板状态：连接/断开/ready/busy
        printf("ECALLBACK_TYPE_FPD_STATUS,recode=%d\n", nlength);
        if (theDemo != NULL)
        {
                CString strMsg = _T("");
                if (nlength <= 0 && nlength >= -11)
                {
                        if (nlength == 0)
                                printf("ECALLBACK_TYPE_FPD_STATUS,Err:网络未连接!\n");
                        else if (nlength == -1)
                                printf("ECALLBACK_TYPE_FPD_STATUS,Err:参数异常!\n");
                        else if (nlength == -2)
                                printf("ECALLBACK_TYPE_FPD_STATUS,Err:准备就绪的描述符数返回失败!\n");
                        else if (nlength == -3)
```

```
            printf("ECALLBACK_TYPE_FPD_STATUS,Err:接收超时!\n");
        else if (nlength == -4)
            printf("ECALLBACK_TYPE_FPD_STATUS,Err:接收失败!\n");
        else if (nlength == -5)
            printf("ECALLBACK_TYPE_FPD_STATUS,Err:端口不可读!\n");
        else if (nlength == -6)
            printf("ECALLBACK_TYPE_FPD_STATUS,network card unusual!\n");
        else if (nlength == -7)
            printf("ECALLBACK_TYPE_FPD_STATUS,network card ok!\n");
        else if (nlength == -8)
            printf("ECALLBACK_TYPE_FPD_STATUS:update Firmware end!\n");
        else if (nlength == -9)
            printf("ECALLBACK_TYPE_FPD_STATUS:光纤已断开!!\n");
        else if (nlength == -10)
            printf("ECALLBACK_TYPE_FPD_STATUS:read ddr failed,try restarting the PCIe driver!\n");
        else /*if (nlength == -11)*/
            printf("ECALLBACK_TYPE_FPD_STATUS:ECALLBACK_TYPE_FPD_STATUS:is not jumb!\n");
        status = (int)FPD_DISCONN_STATUS;
    }
    else if (nlength == FPD_CONN_SUCCESS) { // connect
        printf("ECALLBACK_TYPE_FPD_STATUS,开始监听!\n");
        status = (int)FPD_CONN_SUCCESS;
    }
    else if (nlength == FPD_PREPARE_STATUS) { // ready
        printf("ECALLBACK_TYPE_FPD_STATUS,ready!\n");
        status = (int)FPD_PREPARE_STATUS;
    }
    else if (nlength == FPD_READY_STATUS) { // busy
        printf("ECALLBACK_TYPE_FPD_STATUS,busy!\n");
        status = (int)FPD_READY_STATUS;
    }
    else if (nlength == FPD_DOOFFSET_TEMPLATE) { // prepare
        printf("ECALLBACK_TYPE_FPD_STATUS,prepare!\n");
        status = (int)FPD_DOOFFSET_TEMPLATE;
    }
    else if (nlength == FPD_EXPOSE_STATUS) { // busy expose
        printf("ECALLBACK_TYPE_FPD_STATUS:Exposing!\n");
        status = FPD_EXPOSE_STATUS;
    }
    else if (nlength == FPD_CONTINUE_READY) { // continue ready
        printf("ECALLBACK_TYPE_FPD_STATUS:Continue ready!\n");
        status = FPD_CONTINUE_READY;
    }
    else if (nlength == FPD_DWONLOAD_GAIN) { // download gain template
```

```
            printf("ECALLBACK_TYPE_FPD_STATUS:Download gain template ack!\n");
            status = FPD_DWONLOAD_GAIN;
        }
        else if (nlength == FPD_DWONLOAD_DEFECT) { // download defect template
            printf("ECALLBACK_TYPE_FPD_STATUS:Download defect template ack!\n");
            status = FPD_DWONLOAD_DEFECT;
        }
        else if (nlength == FPD_DWONLOAD_OFFSET) { // download offset template
            printf("ECALLBACK_TYPE_FPD_STATUS:Download offset template ack!\n");
            status = FPD_DWONLOAD_OFFSET;
        }
        else if (nlength == FPD_UPDATE_FIRMARE) { // update firmware
            printf("ECALLBACK_TYPE_FPD_STATUS:Update firmware!\n");
            status = FPD_UPDATE_FIRMARE;
        }
        else if (nlength == FPD_RETRANS_MISS) { // update firmware
            printf("ECALLBACK_TYPE_FPD_STATUS:Retransmission!\n");
            status = FPD_RETRANS_MISS;
        }
        else
            printf("ECALLBACK_TYPE_FPD_STATUS,Err:Other error=%d\n", nlength);

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

            // 触发断开消息
            if (nlength <= 0 && nlength >= -10)
            {
                CFPD_BASE_CFG *pBaseCfg = theDemo->get_base_cfg_ptr(0);
                if (pBaseCfg != NULL)
                {
                    if (pBaseCfg->m_pFpdHand != NULL)
                    {
                        HBI_DisConnectDetector(theDemo->m_pFpd);
                        theDemo->UpdateUI();
                    }
                }
            }
        }
    }
    break;
```

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、固件校正使能等。
通过回调函数返回图像数据地址以及长度，如果需要其他参数请说明。

3.8、模板生成和下载

校正目前分为：offset 校正、gain 校正和 defect 校正。

1》该款平板支持固件做 offset、gain 和 defect 校正；

2》固件可以做 offset 模板，但 gain 和 defect 模板软件做成功后需要下载到固件；

3》做完 offset、gain 和 defect 模板（其中 gain 和 defect 需要下载），启用固件校正使能将会采图后自动实现校正，如果需要原图取消校正使能即可。

快速集成生成模板功能，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 的生成过程和下载做简单说明。

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。

1> Offset 模板

生成模板：

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::OFFSET_TEMPLATE_TYPE;
int ret = HBI_GenerateTemplate(m_pFpd, enumTemplateType);
if (ret != HBI_SUCCSS) {}// 失败
else {} //成功
```

注意：因为是固件做 pre offset 模板，因此不需要再下载到固件。

2> Gain 模板

采集 1 组亮场图，整常高压，毫安秒正常，这里需要高压发生器配合，调用接口前打开高压，等灯管射线稳定后触发命令完成模板生成。

```
EnumIMAGE_ACQ_MODE enumTemplateType = EnumIMAGE_ACQ_MODE::GAIN_TEMPLATE_TYPE;
int ret = HBI_GenerateTemplate(m_pFpd, enumTemplateType);
if (ret != HBI_SUCCSS) {}// 失败
else {} //成功

// 开始下载 Gain 模板
if (0 != HBI_RegProgressCallBack(m_pFpd, DownloadCallBackFun, (void *)this))
{
    printf("err:HBI_RegProgressCallBack failed!\n");
    return;
}
//
if (downloadfile == NULL) downloadfile = new DOWNLOAD_FILE;
if (downloadfile == NULL)
{
    printf("err:malloce downloadfile failed!\n");
    return;
```

```
}
// 参数
downloadfile->emfiletype = GAIN_TMP;
int j = sprintf(downloadfile->filepath, "%s\\hbi_template\\gaina.raw", theDemo->m_path);
downloadfile->filepath[j] = '\0';
int ret = HBI_DownloadTemplate(m_pFpd, downloadfile);
if (ret != HBI_SUCCSS)
	::AfxMessageBox(_T("err:Download gain-template failed!"));
else
	::AfxMessageBox(_T("Download gain-template success!"));
```

3> Defect 模板

采集 3 组亮场图，这里需要高压发生器配合，等灯管射线稳定后触发命令完成模板生成。

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_GenerateTemplate(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_GenerateTemplate(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_GROUP3;
int ret =HBI_GenerateTemplate(m_pFpd, enumTemplateType);
if (ret != HBI_SUCCSS) {}// 失败
else {} //成功

/*
* 模板生成成功后，开始下载模板
*/
// 开始下载 Defect 模板
if (0 != HBI_RegProgressCallBack(m_pFpd, DownloadCallBackFun, (void *)this))
{
	printf("err:HBI_RegProgressCallBack failed!\n");
	return;
}
//
#if 0
if (downloadfile == NULL) downloadfile = new DOWNLOAD_FILE;
if (downloadfile == NULL)
```

```
{
	printf("err:malloce downloadfile failed!\n");
	return;
}
// 参数
if (downloadfile->emfiletype != DEFECT_TMP)downloadfile->emfiletype = DEFECT_TMP;
int j = sprintf(downloadfile->filepath, "%s\\hbi_template\\defect_tp.map",m_path);
downloadfile->filepath[j] = '\0';
int ret = HBI_DownloadTemplate(m_pFpd, downloadfile);
if (ret != HBI_SUCCSS) {
	::AfxMessageBox(_T("err:Download defect template failed!"));
}
else
	::AfxMessageBox(_T("Download defect template success!"));
```

```
#else
int infiletype = 1; // 下载文件类型 0-gain 模板，1-defect 模板，2-offset 模板，其他-不支持
int ret = HBI_DownloadTemplateByType(m_pFpd, infiletype);
if (ret != HBI_SUCCSS)
{
	::AfxMessageBox(_T("err:Download template failed!"));
}
else
{
	::AfxMessageBox(_T("Download template success!"));
}
#endif
```

4> 添加校正使能

做完模板一定要重新设置校正使能。

```
IMAGE_CORRECT_ENABLE *pcorrect = new IMAGE_CORRECT_ENABLE;
if (pcorrect != NULL)
{
	pcorrect->bFeedbackCfg = false;
	pcorrect->ucOffsetCorrection    = 0x03; // 0x00-不做校正，0x01-软件 offset，0x02-固件 poset offset，0x03-固件 preoffset offset,动态平板使用固件 pre offset 即 0x03
	pcorrect->ucGainCorrection      = 0x02;// 0x00-不做校正，0x01-软件 gain，0x02-固件 gain，但需要先下载 gain 模板到平板
	pcorrect->ucDefectCorrection    = 0x02;// 0x00-不做校正，0x01-软件 defect，0x02-固件 defect，但需要先下载 defect 模板到平板
	pcorrect->ucDummyCorrection    = 0x00;// 0x00-不做校正，0x01-暂不支持，0x02-固件暂不支持，目前软件固件都不支持
	ret = HBI_UpdateCorrectEnable(theDoc->m_pFpd, pcorrect);
	if (ret == 0)
```

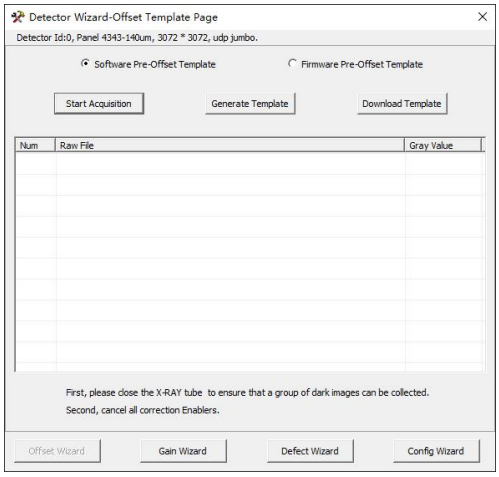
```
        // 成功
    else
        // 失败
    // 释放资源
    delete pcorrect;
    pcorrect = NULL;
}
```

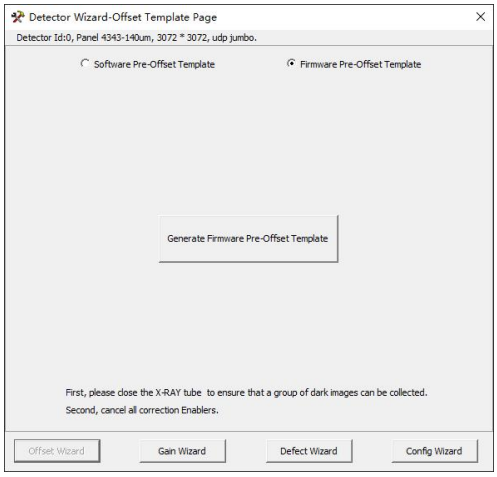
3.9、模板向导（新增）

针对部分客户，在 Dll 中增加了模板向导资源，用户可以直接调用接口打开向导资源窗口，可以生成模板或设置部分参数，结束成后关闭即可，亮场图需要用户高压同步模块配合。向导界面如下所示：

- Offset 模板生成页面

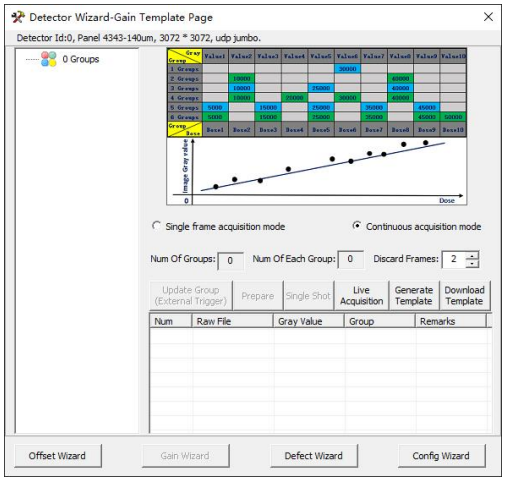
包括生成软件 pre-offset 模板和固件 pre-offset 模板。固件 pre-offset 模板仅支持动态平板，软件 pre-offset 模板支持低帧率静态平板。

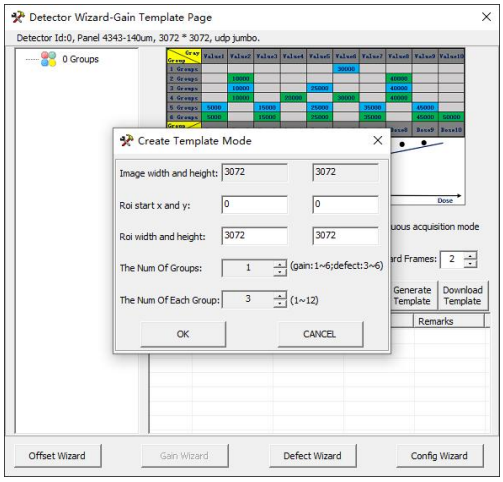




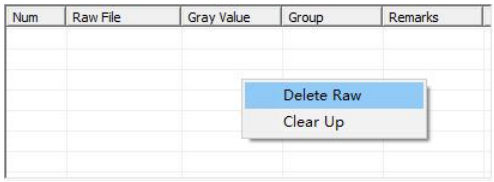
- Gain 模板生成页面

支持不同触发模式下生成 Gain 校正模板，动态平板支持模板下载。生成 Gain 模板过程需要高压配合，正常剂量下至少采集一组亮场图才可以生成 Gain 模板。



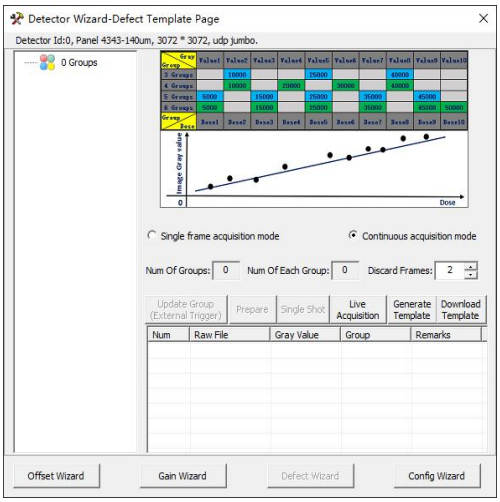


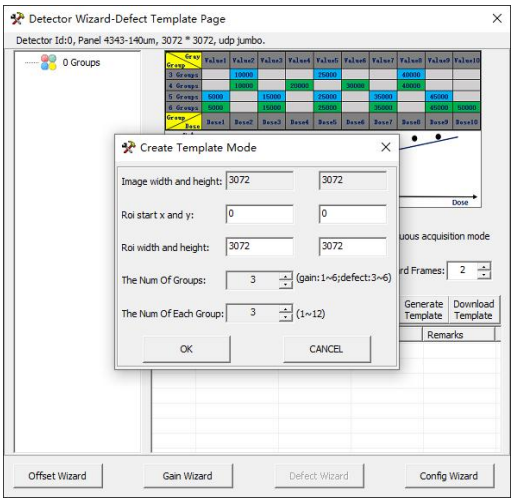
左侧的树控件和右小角的列表控件通过鼠标**“右键”**打开子菜单创建/清空和删除/清空子项。



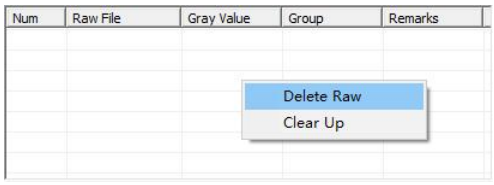
➢ Defect 模板生成页面

支持不同触发模式下生成 Defect 校正模板，动态平板模板支持下载。生成 Defect 模板过程需要高压配合，需要至少采集三组不同剂量下的亮场图才可以生成 Defect 模板。





左侧的树控件和右小角的列表控件通过鼠标**“右键”**打开子菜单创建/清空和删除/清空子项。



➢ Config 页面

主要包括功能：

查看探测器基本信息（平板 ID/类型/分辨率/通讯方式/产品序列号/软件固件版本号/当前图像大小）

设置缩略图模式（部分平板支持）

设置 Binning 类型

设置 PGA 档位

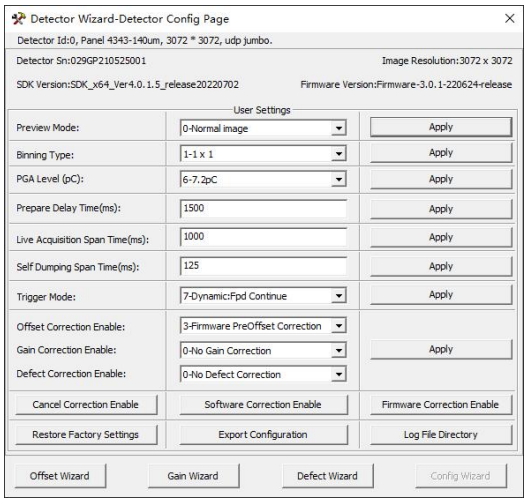
设置 Prepare 延时（控制单帧采集模式）

设置 Live Time 时间（静态平板连续采集时间间隔）

设置 Self-Dumping Time 时间（动态平板连续采集时间间隔，即帧率）

设置触发模式以及设置校正使能状态

恢复出厂设置、导出平板参数以及打开 SDK 日志目录



集成步骤：

```
int ret = HBI_OpenTemplateWizard(m_pFpd); // 打开向导接口函数
if (ret != HBI_SUCCSS)
{
	if (HBI_ERR_NODEVICE == ret)
	{
		::AfxMessageBox(_T("warnning:Dll not initialized!"));
	}
	else if (HBI_ERR_NODEVICE_TRY_CONNECT == ret)
	{
		::AfxMessageBox(_T("warnning:disconnect!"));
	}
	else if (HBI_ERR_INVALID_PARAMS == ret)
	{
		::AfxMessageBox(_T("err:pRomCfg is NULL!"));
	}
	else if (HBI_ERR_OPEN_WIZARD_FAILED == ret)
	{
		::AfxMessageBox(_T("err:open template wizard failed!"));
	}
	else if (HBI_ERR_WIZARD_ALREADY_EXIST == ret)
	{
		::AfxMessageBox(_T("warnning:template wizard already exist!"));
	}
	else
	{
		::AfxMessageBox(_T("err:other error!"));
	}
}
```

具体操作参考 XDiscoveryPro 工具使用手册，与 XDiscoveryPro 中模板向导模块相似。

四、常见问题

依赖库：目前 SDK 使用了 Opencv341，将 Opencv341 的库文件拷贝系统库目录或者 exe 目录下即可，注意 32 位和 64 位库。

有些 Window 系统缺少 vs 库，请安装 vs 环境库。

注意：PCie 驱动：偶尔会出现驱动读取失败，打开设备管理器界面，找到 PCIe 设备驱动，禁用驱动再启用设备驱动。

结束语：由于时间匆忙，文档可能会存在个别问题，望见谅！

附录：

平板序号	平板名称	平板分类	分辨率（w,h）
1	4343-140um	静态、动态、无线	3072，3072

2	3543-140um	静态	2560，3072
3	1613-125um	动态	1248，1024
4	3030-140um	动态	2048，2048
5	2530-85um	静态、动态	2816，3584
6	3025-140um	动态	2048，1792
7	4343-100um	静态、动态	4288，4288
8	2530-75um	静态	3072，3840
9	2121-200um	动态	1024，1024
10	1412-50um	动态	2784，2400
11	0606-50um	动态	1056，1200