综合版本 SDK 开发 DEMO 工程(C#)说明

更新记录

版本号	日期	描述	作者
SDK Ver 4.0.1.5 Doc Ver 1.0	2022.06.15	创建文档	MH.YANG

昊博研发部-软件组

2022-06-15

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+独立网卡+VS2015+C# .Net Framework 4.5.2

设置本地 IP 地址和防火墙，详细见附件昊博 FPD Pro API 接口开发说明书。

二、功能说明

如下图 1-1 所示，分为 4 个主要区域：**图像显示区、窗宽窗位打开保存区、日志显示区以及连接设置采集模板生成下载等控制区。**

图 1-1

1》图像显示区

基于 PictureBox 控件显示 16-bit Raw 文件。可以打开本地 Raw 文件显示或连接平板探测器实时显示，如上图所示。

2》窗宽窗位打开保存区

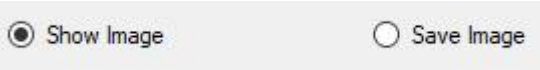
调节图像的窗宽窗位，打开本地 16-bit Raw 和保存当前显示图像。

3》日志显示区

实时监控平板和用户操作，通过打印关键日志信息向用户发聩操作和平板状态。

4》**连接设置采集模板生成下载等控制区**

采集到图像时：当选择“**show Image**”显示图像，“**Save Image**”保存图像。



主要功能包括**连接参数配置、连接平板、断开平板、设置触发模式、设置校正使能状态、获取平板序列号、获取 SDK 和 Firmware 版本号、图像属性、固件参数、单帧采集时间设置、单帧采集命令、连续采集时间设置、连续采集命令、设置 PGA 档位、设置 Binning 类型、快速生成 offset/gain/defect 模板、下载 gain/defect 模板、设置多个控制参数**等。

如下图所示：

序号	下拉框/编辑框/按钮	参数或者输出	接口函数	描述
1	Commnication	0-UDP;以太网千兆网卡，通讯方式是标准 UDP 方式，每包 1kb 1-UDP-Jumbo;通讯方式是标准 UDP Jumbo 方式，每包 9kb 2-PCIe;光口通讯方式 3-Wlan UDP;无线通讯方式	无，连接平板函数 HBI_ConnectDetector 的入参	连接通讯类型
2~3	Remote IP Addr and port	平板探测器的 IP 和端口，仅限网口通讯，光口通讯无效	无，连接平板函数 HBI_ConnectDetector 的入参	通讯参数
4~5	Local IP Addr and port	本地网卡的 IP 和端口，仅限网口通讯，光口通讯无效	无，连接平板函数 HBI_ConnectDetector 的入参	通讯参数，
6	Connect Options	针对动态平板，连接平板时，是否自动生成固件 offset 模板，连接是否做固件 offset 模板，1-固件做 offset 模板，其他不做，且触发模式为 07-Dynamic：continue 触发模式下做固件 pre-offset 模板，其他不做	无，连接平板函数 HBI_ConnectDetector 的入参	连接函数入参
7	Connect	public struct COMM_CFG	HBI_Init 获取实例句柄	连接平板

		{
		public FPD_COMM_TYPE _type;
		// 网口通讯需要设置,PCIe 只要设置类型即可
		[MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
		public byte[] _remoteip;
		[MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
		public byte[] _localip;
		public ushort _loacalPort;
		public ushort _remotePort;
		};
			HBI_RegEventCallBackFun(如果没有注册，请先注册)
			HBI_ConnectDetector	

8	Disconnect	Dll 实例句柄，无符号指针类型	HBI_DisConnectDetector	断开平板
9	FDP Trigger Mode	0-Invalid Trigger Mode;无效触发模式 1-Software Trigger Mode;静态软触发模式 2-Static:Clear Mode;暂不支持 3-Static:Hvg Trigger Mode;静态高压触发模式 4-Static:AED Trigger Mode;静态 AED 触发模式 5-Dynamic:Hvg Sync Mode;动态高压同步模式 6-Dynamic:Fpd Sync Mode;动态平板同步模式 7-Dynamic:Continue Mode;动态 Conitnue 模式	无	触发模式选项
10~13	Offset/Gain/Defect Correction Enable Status	**Offset correct enable：** 0-No Offset Correction;不做 offset 校正 1-Software PreOffset Correction;软件 pre-offset 校正 2-Firmware PostOffset Correction;固件 pre-offset 校正 3-Firmware PreOffset Correction;固件 post-offset 校正 **Gain correct enable：** 0-No Gain Correction;不做 Gain 校正 1-Software Gain Correction;软件 Gain 校正 2-Firmware Gain Correction;固件 Gain 校正 **Detect correct enable：** 0-No Defect Correction;不做 defect 校正 1-Software Defect Correction;软件 defect 校正 2-Firmware Defect Correction;固件 defect 校正 **Dummy correct enable：目前暂不支持** 0-No Dummy Correction;不做 Dummy 校正 1-Software Dummy Correction;软件 Dummy 校正 2-Firmware Dummy Correction;固件 Dummy 校正	无	校正使能状态选项
14	Set Trigger Mode	参数见 9	HBI_UpdateTriggerMode	设置触发模式接口
15~16	Set Correct Enable Get Correct Enable	参数见 10~13	HBI_UpdateCorrectEnable HBI_GetCorrectEnable	设置/获取校正使能状态接口
17	Set Trigger Mode && Correct Enable	参数见 9 和 10~13	HBI_TriggerAndCorrectApplay	设置触发模式和校正使能状态接口
18~	Firmware Version	获取固件版本号	HBI_GetFirmareVerion	获取基本信息

22	Get SDK Version Serial Number Image Property Get Firmware Cfg	获取 sdk 版本号 获取平板序列号 获取图像属性 获取探测器固件参数	HBI_GetSDKVerion HBI_GetFPDSerialNumber HBI_GetImageProperty HBI_GetFpdCfgInfo	
25~27	Set Prepare time Get Prepare time	设置单帧采集模式，通过设置 prepare 延时来控制，时间单位时 ms，》=0，详细见 28~29 单帧采集命令	HBI_SetSinglePrepareTime HBI_GetSinglePrepareTime	单帧采集控制参数
28~29	Single Prepare Single Acquisition	当 prepare 延时为 0 时，发送 HBI_SinglePrepare 后发送 HBI_SingleAcquisition 采集一帧图像，当 prepare 延时.0 时，发送 HBI_SinglePrepare 采集一帧图像	HBI_SinglePrepare HBI_SingleAcquisition	单帧采集命令接口
32~34	Set live acq time interval Get live acq time interval	设置连续采集帧率，通过设置 live time 来控制，时间单位时 ms，》=0，静态和动态接口不一样，注意区分使用，目前可通过通讯类型区分，如下 **静态平板：** 0-UDP;以太网千兆网卡，通讯方式是标准 UDP 方式，每包 1kb 4-Wlan UDP;无线通讯方式 **动态平板：** 1-UDP-Jumbo;通讯方式是标准 UDP Jumbo 方式，每包 9kb 2-PCIe;光口通讯方式 详细见 Demo 工程。	**静态平板** HBI_SetLiveAcquisitionTime HBI_GetLiveAcquisitionTime **动态平板** HBI_SetSelfDumpingTime HBI_GetSelfDumpingTime	连续采集帧率控制参数

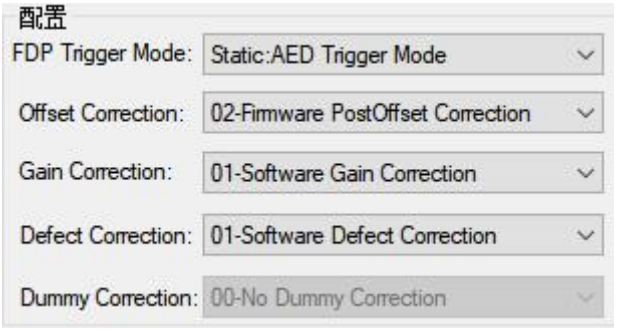
35~36	Live Acquisition
	StopAcquisition
		FPD_AQC_MODE stMode = new FPD_AQC_MODE();
		stMode.eAqccmd= EnumIMAGE_ACQ_CMD.LIVE_ACQ_DEFAULT_TYPE;
		stMode.eLivetype=EnumLIVE_ACQUISITION.ONLY_IMAGE;
		// 1-固件做 offset 模板并上图；2-只上图；3-固件做只做 offset 模板。
		stMode.ngroupno = 0; // 这里默认位 0
		stMode.nAcqnumber = 20; // 0-表示一直采图，20 表示采集 20 帧图结束。这里默认采集 20 帧
		stMode.ndiscard = 0; // 这里默认位 0，不抛弃前几帧图像
		stMode.nframeid = 0; // 这里默认位 0
			HBI_LiveAcquisition
			HBI_StopAcquisition
			当采集帧数为 0 时，需要发送停止连续采集命令，当帧数大于 0 时，完成采集帧数后自动停止采集
				连续采集采集和停止连续采集命令接口

23、30	Set Pga Level	00-Invalid;01-0.6pC;02-1.2pC;03-2.4pC;04-3.6pC;05-4.8pC;06-7.2pC;07-9.6pC;08-LFW(CMOS);09-HFW(CMOS);	HBI_SetPGALevel	设置 PGA 档位，同等剂量越小平板越灵敏，灰度值越大
24、31	Set Binning Type	00-Invalid;01-1 x 1;02-2 x 2;03-3 x 3;04-4 x 4;	HBI_SetBinning	设置 binning 类型
23、24、32、37	Pga+Binning+Fps	PGA 档位：00-Invalid;01-0.6pC;02-1.2pC;03-2.4pC;04-3.6pC;05-4.8pC;06-7.2pC;07-9.6pC;08-LFW(CMOS);09-HFW(CMOS); Binning 类型：00-Invalid;01-1 x 1;02-2 x 2;03-3 x 3;04-4 x 4; Live time：见 32.	HBI_PgaBinningAcqTime	设置 PGA 档位、Binning 类型以及帧率
38	Trigger+Pga+Binning+Fps	触发模式见 9 PGA 档位：00-Invalid;01-0.6pC;02-1.2pC;03-2.4pC;04-3.6pC;05-4.8pC;06-7.2pC;07-9.6pC;08-LFW(CMOS);09-HFW(CMOS); Binning 类型：00-Invalid;01-1 x 1;02-2 x 2;03-3 x 3;04-4 x 4;	HBI_TriggerPgaBinningAcqTime	设置触发模式、PGA 档位、Binning 类型以及帧率

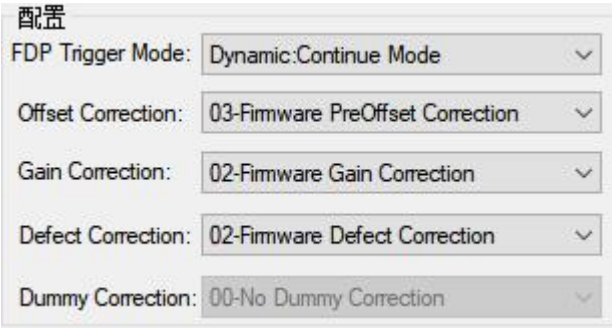
		Live time：见 32.		
39~ 45	Firm Pre-Offset Template Gain Template Defect Template-Group1 Defect Template-Group2 Defect Template-Group3 Soft Pre-Offset Template Download Template	固件生成 pre-offset 模板 采集一组亮场图并生成 gain 模板 采集一组 Defect 亮场图 采集二组 Defect 亮场图 采集三组 Defect 亮场图并生成 defect 模板 软件生成 pre-offset 模板 下载模板文件到探测器固件 **参数如下，详细见 API 文档和 Demo 工程：** // 快速生成矫正模板，用于系统集成 OFFSET_TEMPLATE_TYPE, // 快速生成模板采集类型,连续采集一组暗场图并生成 offset 模板，固件生成模板 GAIN_TEMPLATE_TYPE, // 快速生成模板采集类型,连续采集一组亮场图并生成 gain 模板 DEFECT_TEMPLATE_GROUP1, // 快速生成模板采集类型,连续采集一组亮场图 - defect group1 DEFECT_TEMPLATE_GROUP2, // 快速生成模板采集类型,连续采集一组亮场图 - defect group2 DEFECT_TEMPLATE_GROUP3, // 快速生成模板采集类型,连续采集一组亮场图 - defect group3 and generate template SOFTWARE_OFFSET_TEMPLATE // 快速生成模板采集类型,连续采集一组暗场图 - SDK 生成 offset 模板	HBI_GenerateTemplate HBI_DownloadTemplate（需要注册回调函数 HBI_RegProgressCallBack，监控下载状态进度和结果，详细见 API 文档和 Demo 工程）	生成模板和模板下载，目前软件校正常见于静态低帧率平板，固件校正常见于动态高帧率平板。

三、图像校正

静态平板一般使用软件校正，offset 可以采用 02-Firmware PostOffset 校正效果更佳，默认校正使能如下图所示。



动态平板一般使用固件校正，默认校正使能如下图所示。



例如：
医用或宠物工作站：固件 offset=2、软件 gain=1 和 defect=1 校正，效果比较好。
点料机：固件 offset=3、gain=2 和 defect=2 校正，dummy 校准暂不支持，效果比较好。

四、流程说明

SDK 开发包是基于 C/C++开发标准动态库，包括以下文件
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.《HbFpdDll.h》：导出函数以及说明，具体可参考《昊博 FPD Pro FPD API Programming Reference Ver*.pdf；
b.《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；
c.《HbDllError.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。

在 C#中，可以通过 DllImport 调用 C++ 的非托管 DLL 程序，下面以本工程开发为例来说明。
C/C++头文件中数据类型和接口函数需要转换成 C#类型。

1、头文件
参考 Demo 中 HBI_FPD_DLL.cs 文件，里面包含了接口函数、数据类型（结构体、枚举类等）

```
//Note:fpd communication Type
public enum FPD_COMM_TYPE
{
   UDP_COMM_TYPE       = 0x00,
   UDP_JUMBO_COMM_TYPE = 0x01,
   PCIE_COMM_TYPE      = 0x02,
   WALN_COMM_TYPE      = 0x03
};
。。。 。。。
/*************************************************************
* 函 数 名: HBI_Init
* 功能描述: 初始化动态库
* 参数说明:
*          In: int fpdid - 平板 id，默认为 0
```

```
* 返 回 值：IntPtr
            失败：NULL,成功：非空
* 备     注:
**********************************************************/
[DllImport("HBISDKApi.dll", EntryPoint = "HBI_Init", CharSet = CharSet.Ansi,
CallingConvention = CallingConvention.StdCall)]
public static extern IntPtr HBI_Init(int fpdid=0);
。。。  。。。
```

2、库文件和依赖库

库文件 HBISDKApi.dll

依赖库：opencv_world341.dll

将这两个库文件拷贝到指定目录下，注意 32 为和 64 位。

3、接口调用流程

```
// 1、基本数据类型定义
public bool bSaveImage = false;
public byte nFpdSize = 0;
public int nImageWidth = 0;
public int nImageHeight = 0;
public int nframeId = 0;
Bitmap bitmap;
BitmapData bmapData;
byte[] bmpdata;
Rectangle m_rect = new Rectangle(0, 0, 0, 0);
ImageBuff imgbuff = new ImageBuff();// 共用体 // ushort[] imagetemp;
int imgagesize; // 图像缓冲区大小
//
long windowslevel = 30000;
long windowswidth = 65535;
//
PointF m_pf = new PointF(0, 0);
Font m_font = new Font("Arial", 12);
//
public bool m_bOpen = false;
public static USER_CALLBACK_HANDLE_ENVENT _dUserCallbackHandleEnvent;
public int _ret = 0;
RegCfgInfo m_pLastRegCfg;     // 记录固件所有配置数据  1024 字节的结构体
ImageData imagedata;
ECALLBACK_RAW_INFO m_rawinfo; // 生成模板过程：采集图像保存文件反馈
COMM_CFG commCfg;
IMAGE_CORRECT_ENABLE m_pCorrect = new IMAGE_CORRECT_ENABLE();
DOWNLOAD_FILE m_downfile = new DOWNLOAD_FILE();
//FPD_AQC_MODE m_stMode = new FPD_AQC_MODE();
```

```
public int m_emfiletype = 0;
//
public int nOffset = 0;
public int nGain   = 0;
public int nDefect = 0;
public int nDummy  = 0;
//
public static int nLiveTm      = 0;
public static int nPrepareTm   = 0;
////
public bool btimeout = false;
EnumIMAGE_ACQ_CMD enumTemplateType = EnumIMAGE_ACQ_CMD.OFFSET_TEMPLATE_TYPE;

// 2、初始化 DLL 资源
HBI_FPD_DLL._handel = HBI_FPD_DLL.HBI_Init(0);

// 3、 注册回调函数，用户必须要定义回调函数，否则状态、参数、图像数据无法反馈
// 回调函数见“五、回调函数”
_ret = HBI_FPD_DLL.HBI_RegEventCallBackFun(HBI_FPD_DLL._handel,
_dUserCallbackHandleEnvent, this.Handle);
if (_ret != 0) WriteLog("HBI_RegEventCallBackFun" + _ret.ToString());

// 4、连接平板
// offsettemplate 表示动态平板连接是否自动做 pre-offset 模板
// 1-表示初始化做 offset 模板，非 1 不做 offset 模板
// connect fpd
// 以动态网口巨帧为例，通讯类型 1、udo jumbo，其他参考 Demo 工程
commCfg._type = FPD_COMM_TYPE.UDP_JUMBO_COMM_TYPE;
// IP 地址
commCfg._localip = new byte[16];
byte[] buf = Encoding.UTF8.GetBytes(txtLocalIP.Text);
Buffer.BlockCopy(buf, 0, commCfg._localip, 0, buf.Length);
//
commCfg._remoteip = new byte[16];
buf = Encoding.UTF8.GetBytes(txtRemoteIP.Text);
Buffer.BlockCopy(buf, 0, commCfg._remoteip, 0, buf.Length);
// port
commCfg._loacalPort = ushort.Parse(txtLocalPort.Text.Trim());
commCfg._remotePort = ushort.Parse(txtRemotePort.Text.Trim());
_ret = HBI_FPD_DLL.HBI_ConnectDetector(HBI_FPD_DLL._handel, commCfg, offsettemplate);

// 5、设置触发模式和图像矫正使能
// 常见的触发模式： 1 - 软触发，3 - 高压触发，4 - FreeAED, 7-Dynamic:Continue Mode，
// 其他参考 API 文档或 Demo 工程。
```

```
int _triggerMode = 0;
if (cboxWorkMode.SelectedIndex == 1)
    _triggerMode = 1;
else if (cboxWorkMode.SelectedIndex == 3)
    _triggerMode = 3;
else if (cboxWorkMode.SelectedIndex == 4)
    _triggerMode = 4;
else if (cboxWorkMode.SelectedIndex == 5)
    _triggerMode = 5;
else if (cboxWorkMode.SelectedIndex == 6)
    _triggerMode = 6;
else if (cboxWorkMode.SelectedIndex == 7)
    _triggerMode = 7;
else
    _triggerMode = 7;

//获取矫正使能
m_pCorrect.bFeedbackCfg       = true; // false - ECALLBACK_TYPE_SET_CFG_OK Event
//m_pCorrect.bFeedbackCfg     = false;  // true  - ECALLBACK_TYPE_ROM_UPLOAD Event
m_pCorrect.ucOffsetCorrection = (char)cboxOffsetEnable.SelectedIndex;
m_pCorrect.ucGainCorrection   = (char)cboxGainEnable.SelectedIndex;
m_pCorrect.ucDefectCorrection = (char)cboxDefectEnable.SelectedIndex;
m_pCorrect.ucDummyCorrection  = (char)0/*cboxDummyEnable.SelectedIndex*/; // 暂时不支持
_ret = HBI_FPD_DLL.HBI_TriggerAndCorrectApplay(HBI_FPD_DLL._handel, _triggerMode, ref
m_pCorrect);

// 5、采图或其他设置
// 设置 PGA 档位
//int ret = HBI_FPD_DLL.HBI_SetPGALevel(HBI_FPD_DLL._handel, _pgaLevel);

// 单帧采集
//int ret = HBI_FPD_DLL.HBI_SinglePrepare(HBI_FPD_DLL._handel); //5

// 连续采集
FPD_AQC_MODE stMode = new FPD_AQC_MODE();
stMode.eAqccmd    = EnumIMAGE_ACQ_CMD.LIVE_ACQ_DEFAULT_TYPE;
stMode.eLivetype  = EnumLIVE_ACQUISITION.ONLY_IMAGE;     // 1-固件做 offset 模板并上图；
2-只上图；3-固件做只做 offset 模板。
stMode.ngroupno   = 0;      // 这里默认位 0
stMode.nAcqnumber = 20;     // 0-表示一直采图，20 表示采集 20 帧图结束。这里默认采集 20 帧
stMode.ndiscard   = 0;      // 这里默认位 0，不抛弃前几帧图像
stMode.nframeid   = 0;      // 这里默认位 0

WriteLog("HBI_LiveAcquisition 20frame\n");
```

```
int ret = HBI_FPD_DLL.HBI_LiveAcquisition(HBI_FPD_DLL._handel, stMode);
。。。 。。。

// 最后,DLL 释放资源
// 6、回收资源(包括断开连接和资源释放)
//int ret = HBI_FPD_DLL.HBI_DisConnectDetector(HBI_FPD_DLL._handel);
HBI_FPD_DLL.HBI_Destroy(HBI_FPD_DLL._handel);
```

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_GetSystemConfig 和 HBI_GetFpdCfgInfo：都是读取平板固件参数

HBI_GetSystemConfig 是向固件发请求获取参数，异步函数；HBI_GetFpdCfgInfo 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。

五、回调函数

5.1、主回调函数

```
##############################################
#回调函数及事件说明：
// @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
// @pvParam1:参数 4，配置结构体指针或图像结构体指针
// @nParam2:参数 5，例如 data size(图像数据为图像缓冲区长度)或状态
// @param3:参数 6，例如帧号 frame id
// @param4:参数 7，例如帧率 frame rate 或状态等
// 回调函数结束返回值说明：返回值为整数 0 和 1，1-收到消息，0-未收到消息
如果无返回值或者返回值为 0 或默认为返回 0，SDK 都视为反馈失败，SDK 将记录于日志中，因此上位机收到任何 SDK 的回调消息，请返回 1。
##############################################
// 见 SDK Demo 工程源码
private int hbiMainCallBackFun(IntPtr pContext, int ufpdId, byte eventId, IntPtr ptrParam1,
int nParam2, int nParam3, int nParam4)
{
    int status = 0;
    switch (eventId)
    {
        case (byte)(eCallbackEventCommType.ECALLBACK_TYPE_FPD_STATUS):
            WriteLog(string.Format("ECALLBACK_TYPE_FPD_STATUS,fpdid={0},recode={1}\n",
ufpdId, nParam2));
            if (nParam2 <= 0 && nParam2 >= -11)
```

```
            {
                if (nParam2 == 0)
                    WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,Err:网络未连接!\n");
                else if (nParam2 == -1)
                    WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,Err:参数异常!\n");
                else if (nParam2 == -2)
                    WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,Err:准备就绪的描述符数返回失
败!\n");
                else if (nParam2 == -3)
                    WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,Err:接收超时!\n");
                else if (nParam2 == -4)
                    WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,Err:接收失败!\n");
                else if (nParam2 == -5)
                    WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,Err:端口不可读!\n");
                else if (nParam2 == -6)
                    WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,network card unusual!\n");
                else if (nParam2 == -7)
                    WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,network card ok!\n");
                else if (nParam2 == -8)
                    WriteLog("ECALLBACK_TYPE_NET_ERR_MSG:update Firmware end!\n");
                else if (nParam2 == -9)
                    WriteLog("ECALLBACK_TYPE_FPD_STATUS:光纤已断开!\n");
                else if (nParam2 == -10)
                    WriteLog("ECALLBACK_TYPE_FPD_STATUS:read ddr failed,try restarting
the PCIe driver!\n");
                else /*if (nParam2 == -11)*/
                    WriteLog("ECALLBACK_TYPE_FPD_STATUS:is not jumb!\n");
                status = (int)EFpdStatusType.FPD_DISCONN_STATUS;
            }
            else if (nParam2 == 100)
            { // connect
                WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,开始监听!\n");
                status = (int)EFpdStatusType.FPD_CONN_SUCCESS;
            }
            else if (nParam2 == 1)
            { // connect
                WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,prepare!\n");
                status = (int)EFpdStatusType.FPD_PREPARE_STATUS;
            }
            else if (nParam2 == 2)
            { // ready
                WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,ready!\n");
                status = (int)EFpdStatusType.FPD_READY_STATUS;
            }
            else if (nParam2 == 3)
            { // busy
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Firmware      generates      offset
template!\n");
                status = (int)EFpdStatusType.FPD_DOOFFSET_TEMPLATE;
            }
            else if (nParam2 == 4)
            { // prepare
                WriteLog("ECALLBACK_TYPE_NET_ERR_MSG,busy!\n");
                status = (int)EFpdStatusType.FPD_EXPOSE_STATUS;
            }
            else if (nParam2 == 5)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Continue ready!\n");
                status = (int)EFpdStatusType.FPD_CONTINUE_READY;
            }
            else if (nParam2 == 6)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Download gain template ack!\n");
```

```
                status = (int)EFpdStatusType.FPD_DWONLOAD_GAIN;
            }
            else if (nParam2 == 7)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Download defect template ack!\n");
                status = (int)EFpdStatusType.FPD_DWONLOAD_DEFECT;
            }
            else if (nParam2 == 8)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Download offset template ack!\n");
                status = (int)EFpdStatusType.FPD_DWONLOAD_OFFSET;
            }
            else if (nParam2 == 9)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Update firmware!\n");
                status = (int)EFpdStatusType.FPD_UPDATE_FIRMARE;
            }
            else if (nParam2 == 10)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Retransmission!\n");
                status = (int)EFpdStatusType.FPD_RETRANS_MISS;
            }
            // wireless fpd
            else if (nParam2 == 12)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Sleep State!\n");
                status = (int)EFpdStatusType.FPD_STATUS_SLEEP;
            }
            else if (nParam2 == 13)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:The Last Image!\n");
                status = (int)EFpdStatusType.FPD_DOWNLOAD_TAIL_IMAGE;
            }
            else if (nParam2 == 14)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Download offset template ack!\n");
                status = (int)EFpdStatusType.FPD_STATUS_SLEEP;
            }
            else if (nParam2 == 16)
            { // Last Image
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Maximum number of stored images!");
                status = (int)EFpdStatusType.FPD_EMMC_MAX_NUMBER;
            }
            else if (nParam2 == 17)
            { // prepare
                WriteLog("ECALLBACK_TYPE_FPD_STATUS:Download offset template ack!\n");
                status = (int)EFpdStatusType.FPD_ENDTIME_WARNNING;
            }
            else
                WriteLog(string.Format("ECALLBACK_TYPE_NET_ERR_MSG,Err:other
error={0}\n", nParam2));

            // 异常并断开连接
            if (status != -1)
            {
                // 更新状态信息
                if (nParam2 <= 0 && nParam2 >= -10)
                {
                    //触发断开消息
                    HBI_FPD_DLL.HBI_DisConnectDetector(HBI_FPD_DLL._handel);
                    // 更新平板断开信息
                    btnDisconnect_Click(null, null);
                }
```

```
            }
            break;
        case (byte)(eCallbackEventCommType.ECALLBACK_TYPE_ROM_UPLOAD):
            WriteLog(string.Format("ECALLBACK_TYPE_ROM_UPLOAD\n"));
            if (ptrParam1 != null)
            {
                // 当前反馈固件参数，转化为结构体
                m_pLastRegCfg      =      (RegCfgInfo)Marshal.PtrToStructure(ptrParam1,
typeof(RegCfgInfo));
                // 平板类型
                nFpdSize = (byte)m_pLastRegCfg.m_SysBaseInfo.m_byPanelSize;

                // 平板分辨率
                nImageWidth = m_pLastRegCfg.m_SysBaseInfo.m_sImageWidth;
                nImageHeight = m_pLastRegCfg.m_SysBaseInfo.m_sImageHeight;
                WriteLog(string.Format("\tnFpdSize={0},Image    width={0},hight={1}\n",
nFpdSize, nImageWidth, nImageHeight));

                // 打印参数,同步参数比较慢 ？？？
                //#if 1
                 //      PrintFirmwrae(); // 打印结构体信息
                //#endif
            }
            break;
        case (byte)(eCallbackEventCommType.ECALLBACK_TYPE_SET_CFG_OK):
            WriteLog("ECALLBACK_TYPE_SET_CFG_OK!\n");
            break;
        //----------------图像-------------------
        case (byte)(eCallbackEventCommType.ECALLBACK_TYPE_SINGLE_IMAGE):
        case (byte)(eCallbackEventCommType.ECALLBACK_TYPE_MULTIPLE_IMAGE):
            WriteLog("success,ECALLBACK_TYPE_MULTIPLE_IMAGE                              or
ECALLBACK_TYPE_SINGLE_IMAGE!\n");
            if (ptrParam1 == null)
            {
                WriteLog("err:ptrParam1 is null!\n");
                return 1;
            }

            // 当前反馈固件参数，转化为结构体 add by mhyang 20220402
            imagedata = (ImageData)Marshal.PtrToStructure(ptrParam1, typeof(ImageData));

            // add by mhyang 20220615
            if (bSaveImage)
                SaveImage();
            else
                ShowImage();
            break;
        case (byte)(eCallbackEventCommType.ECALLBACK_TYPE_GENERATE_TEMPLATE):
            {
                if (nParam2 == 0)
                {
                    WriteLog("success,ECALLBACK_TYPE_MULTIPLE_IMAGE                    or
ECALLBACK_TYPE_SINGLE_IMAGE!\n");
                }
                else if (nParam2 == 1)
                {

WriteLog(string.Format("ECALLBACK_TEMPLATE_INVALVE_PARAM,fpdid={0},recode={1}\n",
ufpdId, nParam3));
                }
                else if (nParam2 == 2)
                {
```

```
WriteLog(string.Format("ECALLBACK_TEMPLATE_MALLOC_FAILED,fpdid={0},recode={1}\n",
ufpdId, nParam3));
                }
                else if (nParam2 == 3)
                {

WriteLog(string.Format("ECALLBACK_TEMPLATE_SEND_FAILED,fpdid={0},recode={1}\n", ufpdId,
nParam3));
                }
                else if (nParam2 == 4)
                {

WriteLog(string.Format("ECALLBACK_TEMPLATE_STATUS_ABORMAL,fpdid={0},recode={1}\n",
ufpdId, nParam3));
                }
                else if (nParam2 == 5)
                {

WriteLog(string.Format("ECALLBACK_TEMPLATE_FRAME_NUM,fpdid={0},recode={1}\n",    ufpdId,
nParam3));
                }
                else if (nParam2 == 6)
                {

WriteLog(string.Format("ECALLBACK_TEMPLATE_TIMEOUT,fpdid={0},recode={1}\n",      ufpdId,
nParam3));
                }
                else if (nParam2 == 7)
                {
                    m_rawinfo  =  (ECALLBACK_RAW_INFO)Marshal.PtrToStructure(ptrParam1,
typeof(ECALLBACK_RAW_INFO));
                    WriteLog("ECALLBACK_TEMPLATE_MEAN:dMean="                          +
m_rawinfo.dMean.ToString());
                    WriteLog("ECALLBACK_TEMPLATE_MEAN:dMean="                          +
m_rawinfo.szRawName.ToString());
                }
                else if (nParam2 == 8)
                {
                    if (nParam3 == (byte)(emUPLOAD_FILE_TYPE.OFFSET_TMP))
                        WriteLog("ECALLBACK_TEMPLATE_GENERATE:OFFSET_TMP!\n");
                    else if (nParam3 == (byte)(emUPLOAD_FILE_TYPE.GAIN_TMP))
                        WriteLog("ECALLBACK_TEMPLATE_GENERATE:GAIN_TMP\n");
                    else if (nParam3 == (byte)(emUPLOAD_FILE_TYPE.DEFECT_TMP))

WriteLog(string.Format("ECALLBACK_TEMPLATE_GENERATE:DEFECT_TMP,bad         point={0}\n",
nParam4));
                    else

WriteLog(string.Format("ECALLBACK_TEMPLATE_GENERATE,fpdid={0},recode={1}\n",     ufpdId,
nParam3));
                }
                else if (nParam2 == 9)
                {
                    WriteLog(string.Format("ECALLBACK_TEMPLATE_RESULT={0}\n",
nParam3));
                }
                else
                {// other

WriteLog(string.Format("ECALLBACK_TEMPLATE_GENERATE,other:fpdid={0},recode={1}\n",
ufpdId, nParam3));
                }
                break;
```

```
            }
        case (byte)(eCallbackEventCommType.ECALLBACK_OVERLAY_16BIT_IMAGE):
            break;
        case (byte)(eCallbackEventCommType.ECALLBACK_OVERLAY_32BIT_IMAGE):
            break;
        case (byte)(eCallbackEventCommType.ECALLBACK_TYPE_FILE_NOTEXIST):
            break;
#region
        case (byte)(eCallbackEventCommType.ECALLBACK_TYPE_THREAD_EVENT):
            if (nParam2 == 100)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,start recv data!\n");
            else if (nParam2 == 101)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,end recv data!\n");
            else if (nParam2 == 104)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,Packet  Retransmission:start  recv
data!\n");
            else if (nParam2 == 105)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,Frame  Retransmission:start  recv
data!\n");
            else if (nParam2 == 106)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,Frame loss retransmission over,end
recv data!\n");
            else if (nParam2 == 107)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,image  buff  is  null:end  recv
data!\n");
            else if (nParam2 == 108)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,Generate  Offset  Template:start
thread!\n");
            else if (nParam2 == 109)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,Generate   Offset   Template:end
thread!\n");
            else if (nParam2 == 110)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,Generate   Gain   Template:start
thread!\n");
            else if (nParam2 == 111)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,Generate    Gain    Template:end
thread!\n");
            else if (nParam2 == 112)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,offset calibrate:success!\n");
            else if (nParam2 == 113)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,offset calibrate:failed!\n");
            else if (nParam2 == 114)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,gain calibrate:success!\n");
            else if (nParam2 == 115)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,gain calibrate:failed!\n");
            else if (nParam2 == 116)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,defect calibrate:success!\n");
            else if (nParam2 == 117)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,defect calibrate:failed!\n");
            else if (nParam2 == 118)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,InitGainTemplate:failed!\n");
            else if (nParam2 == 119)
                WriteLog("ECALLBACK_TYPE_THREAD_EVENT,firmare                    offset
calibrate:success!\n");
            else
                WriteLog(string.Format("ECALLBACK_TYPE_THREAD_EVENT,Err:  未  知  错  误
[{0}]\n", nParam2));
            break;
#endregion
        default:
          //
WriteLog(string.Format("ECALLBACK_TYPE_THREAD_EVENT,ECALLBACK_TYPE_INVALVE,command=[{0}]
\n", cmd));
```

```
            break;
    }
    return 1;
}
```

5.2、模板下载回调函数

```
/*-----------------------------------------------
// Notice: call back function
// @USER_CALLBACK_HANDLE_PROCESS
// @cmd:enum eAnswerCallbackCommType
// @retcode: return code
// @inContext: position machine object point
-----------------------------------------------*/
// dwnload template callcallback function
private int DownloadCallBack(byte command, int code, IntPtr pContext)
{
    switch (command)
    {
        case        (byte)(eCallbackUpdateFirmwareStatus.ECALLBACK_UPDATE_STATUS_START):
// 开始 初始化
            WriteLog("DownloadCallBack   :    ECALLBACK_UPDATE_STATUS_START   :    start
download\n");
            break;
        case     (byte)(eCallbackUpdateFirmwareStatus.ECALLBACK_UPDATE_STATUS_PROGRESS):
// 进度 code: 百分比（0~100）
            WriteLog(string.Format("DownloadCallBack                                    :
ECALLBACK_UPDATE_STATUS_PROGRESS: %% {0}\n", code));
            break;
        case       (byte)(eCallbackUpdateFirmwareStatus.ECALLBACK_UPDATE_STATUS_RESULT):
// update result and error
            if ((0 <= code) && (code <= 6))
            {
                if (code == 0)
                {   // 下载 offset 模板
                    WriteLog("DownloadCallBack: ECALLBACK_UPDATE_STATUS_RESULT: download
offset template!\n");
                }
                else if (code == 1)
                {  // 下载 gain 模板
                    WriteLog("DownloadCallBack: ECALLBACK_UPDATE_STATUS_RESULT: download
gain template!\n");
                }
                else if (code == 2)
                {  // 下载 defect 模板
                    WriteLog("DownloadCallBack: ECALLBACK_UPDATE_STATUS_RESULT: download
defect template!\n");
                }
                else if (code == 3)
                {  // offset 模板上传完成
                    WriteLog("DownloadCallBack: ECALLBACK_UPDATE_STATUS_RESULT: download
offset finish!\n");
                }
                else if (code == 4)
                {  // gain 模板上传完成
                    WriteLog("DownloadCallBack: ECALLBACK_UPDATE_STATUS_RESULT: download
gain finish!\n");
                }
                else if (code == 5)
                {  // defect 模板上传完成
                    WriteLog("DownloadCallBack: ECALLBACK_UPDATE_STATUS_RESULT: download
```

```
defect finish!\n");
                }
                else/* if (code == 6)*/
                {  // 模板上传完成
                    WriteLog("DownloadCallBack：ECALLBACK_UPDATE_STATUS_RESULT：Download
finish and sucess!!\n");
                }
            }
            else // 失败
            {
                if (code == -1)
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT ： wait event other
error!\n");
                }
                else if (code == -2)
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT：timeout!\n");
                }
                else if (code == -3)
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT ： downlod offset
failed!\n");
                }
                else if (code == -4)
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT ： downlod gain
failed!\n");
                }
                else if (code == -5)
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT ： downlod defect
failed!\n");
                }
                else if (code == -6)
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT：Download failed!\n");
                }
                else if (code == -7)
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT ： read offset
failed!!\n");
                }
                else if (code == -8)
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT：read gain failed!!\n");
                }
                else if (code == -9)
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT ： read defect
failed!!\n");
                }
                else
                {
                    WriteLog("err:ECALLBACK_UPDATE_STATUS_RESULT：unknow error!\n");
                }
            }
            break;
        default: // unusual
            WriteLog(string.Format("DownloadCallBack,retcode:[{0}]\n", code));
            break;
    }
    return 1;
```

}

六、生成模板和模板下载

快速生成模板：主要包括生成 Offset、Gain 和 Defect 模板，下面**剂量值仅为参考值**。

✓ Offset 模板

需要采集暗场图像，只需要采集一组图像；

步骤：

Offset 模板一般需要每天重做一次(包括不存在 offset 模板)。

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，软件端完成 offset 模板并在本地保存一张 offset 模板。

✓ 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》调节剂量采集第一组亮场；

2》调节剂量采集第二组亮场；

3》调节剂量采集完第三组亮场图，将自动生成 defect 模板，并保存在本地；

具体具体接口和流程如下

6.1、生成模板

1> Offset 模板

```
// 第一步：一组暗场(关闭 X 光高压球管)-发送采集命令
// 动态平板-固件生成 pre-offset 模板
enumTemplateType = EnumIMAGE_ACQ_CMD.OFFSET_TEMPLATE_TYPE;
int ret = HBI_FPD_DLL.HBI_GenerateTemplate(HBI_FPD_DLL._handel, enumTemplateType);

//// 静态平板-软件生成 pre-offset 模板
////enumTemplateType = EnumIMAGE_ACQ_CMD.SOFTWARE_OFFSET_TEMPLATE;//软件生成 pre-offset 模板
////int ret = HBI_FPD_DLL.HBI_GenerateTemplate(HBI_FPD_DLL._handel, enumTemplateType);
```

2> Gain 模板

// 第一步：一组亮场(剂量要求：正常高压和电流)-发送采集命令

// 高压的剂量一般为正常采图的剂量即可，等高压到达稳定值后开始调用生成接口直到采图完成结束。

```
enumTemplateType = EnumIMAGE_ACQ_CMD.GAIN_TEMPLATE_TYPE;
int ret = HBI_FPD_DLL.HBI_GenerateTemplate(HBI_FPD_DLL._handel, enumTemplateType);
```

// 以下动态平板校正前提是把模板下载到平板固件

// 第二步：注册回调函数

```
ret = HBI_FPD_DLL.HBI_RegProgressCallBack(HBI_FPD_DLL._handel, DownloadCallBack, this.Handle);
```

// 第三步：将 gain 模板下载到固件

```
m_emfiletype = 0;
HBI_FPD_DLL.HBI_DownloadTemplateByType(HBI_FPD_DLL._handel, m_emfiletype);
```

// 第四步：更新矫正使能

```
m_pCorrect.bFeedbackCfg = true; // true - ECALLBACK_TYPE_ROM_UPLOAD Event,false - ECALLBACK_TYPE_SET_CFG_OK Event
m_pCorrect.ucOffsetCorrection = (char)3;
m_pCorrect.ucGainCorrection = (char)2;
m_pCorrect.ucDefectCorrection = (char)0;
m_pCorrect.ucDummyCorrection = (char)0; // 暂时不支持
ret = HBI_FPD_DLL.HBI_UpdateCorrectEnable(HBI_FPD_DLL._handel, ref m_pCorrect);
```

3> Defect 模板

```
// 第一步：第一组亮场(剂量要求：正常高压，毫安秒调节正常的 10%)-发送采集命令
enumTemplateType = EnumIMAGE_ACQ_CMD.DEFECT_TEMPLATE_GROUP1;
_ret = HBI_FPD_DLL.HBI_GenerateTemplate(HBI_FPD_DLL._handel, enumTemplateType);

// 第二步：第二组亮场(剂量要求：正常高压，毫安秒调节正常的 50%)-发送采集命令
enumTemplateType = EnumIMAGE_ACQ_CMD.DEFECT_TEMPLATE_GROUP2;
_ret = HBI_FPD_DLL.HBI_GenerateTemplate(HBI_FPD_DLL._handel, enumTemplateType);

// 第三步：第二组亮场(剂量要求：正常高压，毫安秒调节正常的 100%)-发送采集命令
enumTemplateType = EnumIMAGE_ACQ_CMD.DEFECT_TEMPLATE_GROUP3;
_ret = HBI_FPD_DLL.HBI_GenerateTemplate(HBI_FPD_DLL._handel, enumTemplateType);
```

// 以下动态平板校正前提是把模板下载到平板固件

// 第四步：注册回调函数

```
ret = HBI_FPD_DLL.HBI_RegProgressCallBack(HBI_FPD_DLL._handel, DownloadCallBack, this.Handle);
```

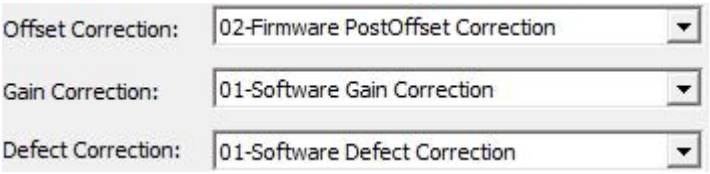
// 第五步：将 defect 模板下载到固件
m_emfiletype = 1;
HBI_FPD_DLL.HBI_DownloadTemplateByType(HBI_FPD_DLL._handel, m_emfiletype);

// 第四步：更新矫正使能
m_pCorrect.bFeedbackCfg = true; // true - ECALLBACK_TYPE_ROM_UPLOAD Event,false - ECALLBACK_TYPE_SET_CFG_OK Event
m_pCorrect.ucOffsetCorrection = (char)3;
m_pCorrect.ucGainCorrection = (char)2;
m_pCorrect.ucDefectCorrection = (char)2;
m_pCorrect.ucDummyCorrection = (char)0; // 暂时不支持
ret = HBI_FPD_DLL.HBI_UpdateCorrectEnable(HBI_FPD_DLL._handel, ref m_pCorrect);

具体请参考 Demo 源码例子工程。

3、校正使能

医用或宠物工作站：固件 offset、软件 gain 和 defect 校正，效果比较好。



点料机：软件 offset、gain 和 defect 校正，效果比较好。

以点料机为例：

m_pCorrect.bFeedbackCfg = true; // true - ECALLBACK_TYPE_ROM_UPLOAD Event,false - ECALLBACK_TYPE_SET_CFG_OK Event
m_pCorrect.ucOffsetCorrection = (char)3;
m_pCorrect.ucGainCorrection = (char)2;
m_pCorrect.ucDefectCorrection = (char)2;
m_pCorrect.ucDummyCorrection = (char)0; // 暂时不支持
ret = HBI_FPD_DLL.HBI_UpdateCorrectEnable(HBI_FPD_DLL._handel, ref m_pCorrect);

6.2、下载模板

// 第一步：注册回调函数
ret = HBI_FPD_DLL.HBI_RegProgressCallBack(HBI_FPD_DLL._handel, DownloadCallBack, this.Handle);

// 第二步：将 defect 模板下载到固件
m_emfiletype = 1; // 0-gain 模板，1-defect 模板，2-offset 模板
HBI_FPD_DLL.HBI_DownloadTemplateByType(HBI_FPD_DLL._handel, m_emfiletype);

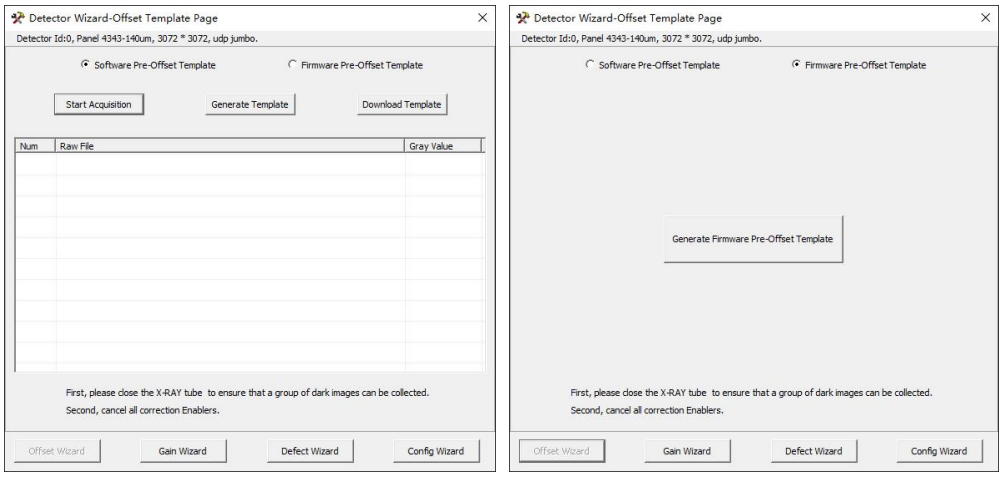
具体请参考 SDK Demo 工程和 API 文档。

七、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：

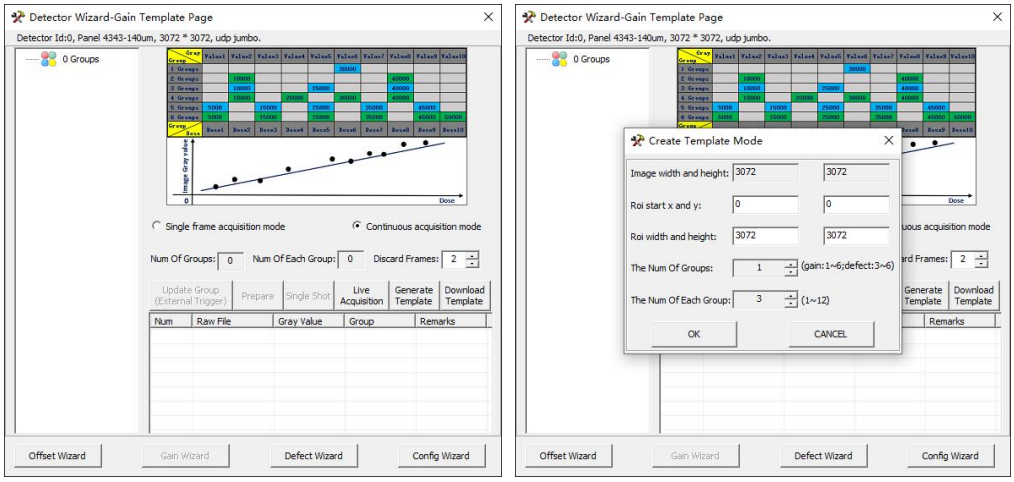
- Offset 模板生成页面

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。

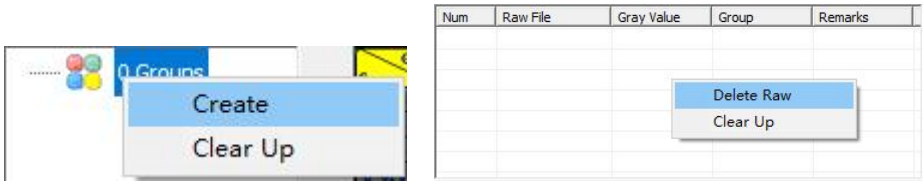


- Gain 模板生成页面

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。

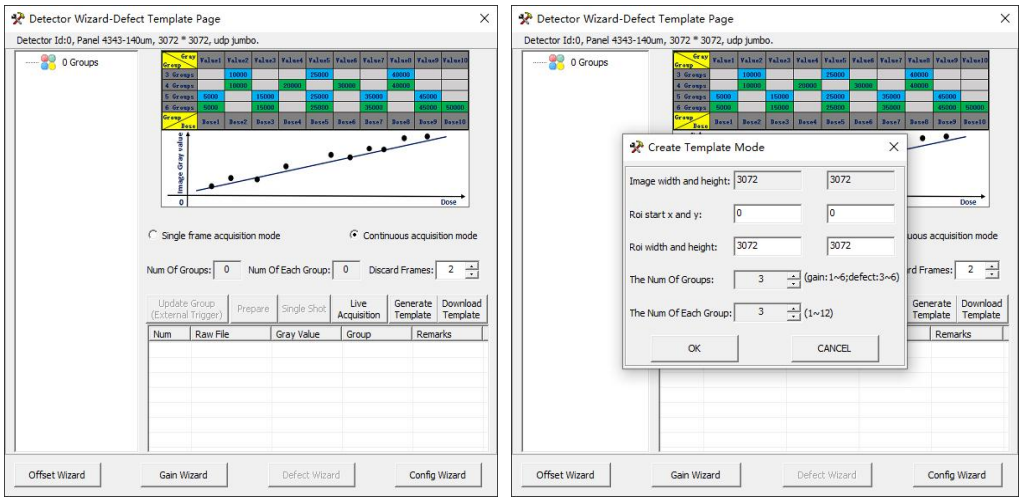


左侧的树控件和右小角的列表控件通过鼠标**“右键”**打开子菜单创建/清空和删除/清空子项。

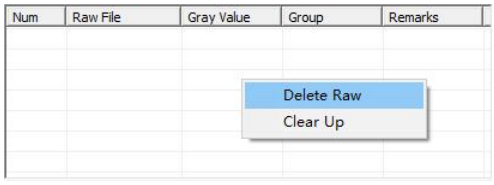


- Defect 模板生成页面

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

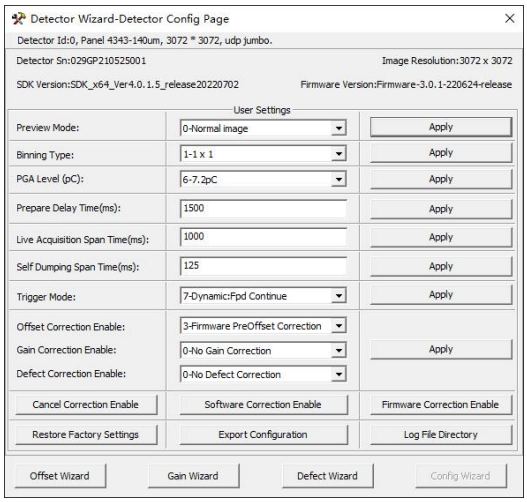
设置 Prepare 延时（控制单帧采集模式）

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_FPD_DLL.HBI_OpenTemplateWizard(HBI_FPD_DLL._handel);
if (ret != 0)
{
    if (8007 == ret)
    {
            System.Windows.Forms.MessageBox.Show("warnning:Dll  not  initialized!"  +
ret.ToString());
    }
    else if (8008 == ret)
    {
        System.Windows.Forms.MessageBox.Show("warnning:disconnect!" + ret.ToString());
    }
    else if (8002 == ret)
    {
        System.Windows.Forms.MessageBox.Show("err:pRomCfg is NULL!" + ret.ToString());
    }
    else if (8036 == ret)
    {
          System.Windows.Forms.MessageBox.Show("err:open  template  wizard  failed!"  +
ret.ToString());
    }
    else if (8037 == ret)
    {
        System.Windows.Forms.MessageBox.Show("warnning:template wizard already exist!" +
ret.ToString());
    }
    else
    {
        System.Windows.Forms.MessageBox.Show("err:other error!" + ret.ToString());
    }
}
else
{
    WriteLog("HBI_OpenTemplateWizard success!");
}
```

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

八、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。

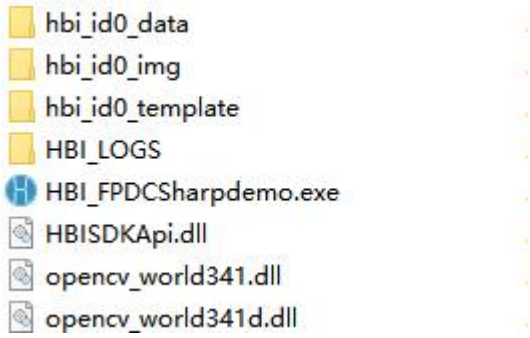
vcredist_2015_x64.exe 或 vcredist_2015_x86.exe

结束语：由于时间匆忙，文档可能会存在个别问题，望见谅！

附录一：

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

附录二：



，其中

hbi_id0_data：生成模板中间图像文件
hbi_id0_img：图像文件保存目录
hbi_id0_template：模板文件目录
HBI_LOGS：日志文件目录
HBI_FPDCSharpdemo.exe：Demo 生成文件
HBISDKApi.dll：昊博 Dll 文件
opencv_world341.dll：dll 文件依赖库，release 版本
opencv_world341d.dll：dll 文件依赖库，debug 版本