	密级：机密 北京智影技术有限公司 开发文档	文件编号： 版　　本：版本 页　　码：共 51 页
项目名称： 便携式彩色多普勒超声诊断系统 文件名称： **CIES_SDK** 适用范围：		
编制：武龙 日期：2021/4/09	审核： 日期：	批准： 日期：

修　订　记　录

版本	修订内容概述	修订人	修订日期
1.0	创建	武龙	2020/9/28
1.01	增加 PCB_PCBA_Info 类，在 USDeviceInfo 类中在增加 PCB_PCBA_Info 内容。 FrameDataListener 类中增加新的监听接口，用来监听探头插拔消息。探头在位消息通过在位信号进行判断。 ProbeInfo 类增加发射显示频率值，B 模式切换高中低频时显示出频率，C 模式切换高速低速是显示出频率	武龙	2020/12/04
1.02	ColorImageEngineServer 类增加设置 D M 模式参数接口，取样门、取样线设置参数接口，获取实时数据和历史数据接口。增加获取当前参数下	武龙	2021/04/09

	焦点范围的接口。 ShowFrame 文件中增加 D M 模式数据帧和相关参数 IESParamter 文件中增加 D M 模式设置参数类；修改 B 参数焦区参数设置，从近、中、远、全域，修改成焦点 1—6、全域；ProbeInfo 类则你感觉 D 的偏转角度；ScanMode 类增加 D M 模式		
1.03	1. B_ImageParam 增加帧增强参数 2. ProbeInfo 增加帧增强参数获取 3. ImageDataInfo 增加 C 模式发射起始线结束线 4. D_PWDataInfo 增加血管自动角度和管径测量参数返回 5.增加多个自动测量功能：自动管径测量，自动 TGC，D 模式自动取样门设置，D 模式自动 PRF 和基线		2021/08/12
1.04	1.更新算法，支持 USB 探头 2.更新参数，支持 8/16/32 发射通道 USB 探头		2021/09/26
1.05	1.修改 SDK，修正 M 模式刷新速度发生错误的问题。		2021/10/08
1.06	1.更新算法，增加 B 模式自定义灰阶第 9 档。优化图像后处理。 2.更新参数。增加新探头		2021/11/18

CIES_SDK

目录

概述

本文档对 CIES_SDK 目录结构进行介绍，对类及类方法进行详细说明，并提供示例 Demo 程序。

目录结构简介

SDK 目录结构如下所示

conf：

-- Log.cfg ： 日志配置文件

doc：

-- CIES_SDK ： SDK 说明文档

include：

-- CIESParamter.h ： 图像引擎参数头问题

-- ColorImageEngineServer.h ： 图像引擎头文件

--FrameDataListener.h ： 监听接口

--Native.h ： 监听子类

--Native.cpp ： 监听子类实现

--DataSourceInterface.h ： 数据源接口

--DataSource.h ： 数据源接口子类

lib：

-- ColorImageEngineServer.lib ： 图像引擎 SDK lib

-- ColorImageEngineServer.dll ： 图像引擎 SDK dll

--… … 其他文件需要和 DLL 文件在同一个目录下

param：

-- ProbeConfig.xml ： 探头配置文件

-- C5-2 ： 探头文件夹

-- ExamConfig.xml ： 检查模式配置文件

-- ABD ： ABD 检查模式参数目录

-- … … ： 其他检查模式参数目录

-- C5-2_ResetParameter ： 探头参数预制档位文件夹

-- ABD ： ABD 检查模式参数预制档位目录

usbDriver ： windowsUSB 驱动

conf

Log.cfg

SDK 的 log 配置文件，需要放在 dll 上级 conf 目录中。例：..\conf。

param

ProbeConfig.xml

探头配置文件:

```
<?xml version="1.0"?>
<ProbeConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Version="1.0">
  <ProbeInfoList>
    <ProbeInfo>
      <ProberName>C5-2</ProberName>
      <ProberFrequency>3.5MHz</ProberFrequency>
      <ProberShowName>C5-2</ProberShowName>
      <ProberCanShow>false</ProberCanShow>
      <IsSupportPW>true</IsSupportPW>      </ProbeInfo>
  </ProbeInfoList>
</ProbeConfig>
```

描述了当前 param 下包含什么探头参数和探头对应的频率
此文件用于对探头列表的解析，方便开发者对探头及探头信息获取，也可以自定义格式替换。

C5-2

- “C5-2 “ 是探头名称，C5-2 文件夹下是该探头的参数目录。
- ExamConfig.xml ：检查模式配置文件

```
<?xml version="1.0"?>
<ExamConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Version="1.0">
  <ExamInfoList>
    <ExamInfo>
      <ExamName_ZH>腹部</ExamName_ZH>
      <ExamName_EN>ABD</ExamName_EN>
      <ExamNameCanShow>true</ExamNameCanShow>
      <ExamOrder>1</ExamOrder>
      <ExamName_Alias />
      <IsDefaultExam>false</IsDefaultExam>
    </ExamInfo>
  </ExamInfoList>
</ExamConfig>
```

存储当前探头对应的检查模式，中英文显示。也可以在自定义其他格式

C5-2_ResetParameter

```
<?xml version="1.0"?>
<ImageParamConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Version="1.0">
  <GainLevel>50</GainLevel>
  <DepthLevel>6</DepthLevel>
  <DynamicRange>70</DynamicRange>
  <IsImageEnhancement>true</IsImageEnhancement>
  <SRILevel>4</SRILevel>
  <CorrelationLevel>2</CorrelationLevel>
  <GrayLevel>4</GrayLevel>
  <PseudocolorLevel>0</PseudocolorLevel>
  <L_R>0</L_R>
  <T_B>0</T_B>
  <Harmonic>1</Harmonic>
  <Frequency>0</Frequency>
  <FocusArea>3</FocusArea>
</ImageParamConfig>
```

存储当前探头对应的检查模式的 B\C\D\M 预制参数档位或数值。也可以在自定义其他格式

UL10-5E-8/UL10-5E-16/UL10-5E-32

探头参数文件名以“U”开头的是 USB 探头参数。
探头参数文件名以以“-X”结尾的代表该参数是 X 发射通道探头的参数。
当前 X 取值范围：8、16、32。

usbDriver

windows 下 usb 驱动程序。
以管理员权限运行 InstallDriver.bat 就会自动安装 usb 驱动程序。
系统要求是 win8.1，windows10

lib

存储 x86 版本库

-- ColorImageEngineServer.lib ： 图像引擎 SDK lib
-- ColorImageEngineServer.dll ： 图像引擎 SDK dll

lib 内其他文件需要和 DLL 文件在同一个目录下

lib_x64

存储 x64 版本库

-- ColorImageEngineServer.lib ： 图像引擎 SDK lib

-- ColorImageEngineServer.dll ： 图像引擎 SDK dll

lib 内其他文件需要和 DLL 文件在同一个目录下

API 说明

ColorImageEngineServer Class

头文件

ColorImageEngineServer.h

说明

ColorImageEngineServer 类是图像引擎对外的主要类。

- ColorImageEngineServer 类使用 StartImageEngine 和 StopImageEngine 方法对图像引擎进行启动和停止控制。
- ColorImageEngineServer 类使用 IsUSBOpen 方法对 USB 连接状态进行判断。
- ColorImageEngineServer 类使用 Freeze 和 UnFreeze 方法来冻结和解冻图像。
- ColorImageEngineServer 类使用 SetUseImageHistoryCache 方法来设置是否适用图像缓存，和设置图像缓存长度。
- ColorImageEngineServer 类使用 SetFrameDataListener 方法来监听硬件数据。
- ColorImageEngineServer 类使用 GetImageDisplayData、GetImageHistoryDisplayData 和 GetImageHistoryDisplayDataCount 方法获取时候图像数据和历史图像数据。
- ColorImageEngineServer 类使用 SetCheckModeParamPath_And_ScanMode 、SetImageParam_B、SetImageParam_C、SetImageParam_D_PW、SetImageParam_M 方法来设置需要的参数文件路径和 B\C 的参数。
- ColorImageEngineServer 类使用 GetColorMap 、GetColorMap_B 、GetColorMap_C 、GetColorMap_D、GetColorMap_M 方法来获取彩色图谱。
- ColorImageEngineServer 类使用 GetCurrentFrameRate 方法来获取当前参数下理论帧率。
- ColorImageEngineServer 类使用 GetProberInfo 方法来获取探头参数和深度列表。
- ColorImageEngineServer 类使用 GetD_PWInfo 方法来获取探头 D 模式参数。
- ColorImageEngineServer 类使用 GetHardWareVersion 方法来获取硬件版本。
- ColorImageEngineServer 类使用 GetVersion 方法获取当前 SDK 版本。
- ColorImageEngineServer 类使用 GetImageWidthPixels 方法获取 B 图像宽度像素(不包括黑边)。
- ColorImageEngineServer 类使用 GetHardWareInfo 方法获取硬件消息。
- ColorImageEngineServer 类使用 GetImageHeightPixels 方法获取 B 图像高度像素(不包括黑边)。
- ColorImageEngineServer 类使用 GetCurrentFocusArea 方法获取当前焦区范围 废弃接口。
- ColorImageEngineServer 类使用 GetCurrentFocusArray 方法获取当前焦区范围。
- ColorImageEngineServer 类使用 GetCurrentFocusArrayByParamPath 方法获取任意探头的焦区范围。
- ColorImageEngineServer 类使用 SetFocusArea 方法设置任意焦点。
- ColorImageEngineServer 类使用 SetCSamplingFrameParam_Quadrangle 方法设置线阵取样窗信息。
- ColorImageEngineServer 类使用 Get_C_PRFList_Quadrangle 方法获取线阵 PRF 信息。
- ColorImageEngineServer 类使用 SetCSamplingFrameParam_Sector 方法设置凸阵和相控阵

取样窗信息

- ColorImageEngineServer 类使用 Get_C_PRFList_Sector 方法获取凸阵和相控阵 PRF 信息。
- ColorImageEngineServer 类使用 SetD_PWSamplingGateParam_Line 方法设置线阵取样门信息。
- ColorImageEngineServer 类使用 Get_D_PRFList_Line 方法获取线阵 PRF 信息。
- ColorImageEngineServer 类使用 SetD_PWSamplingGateParam_Convex 方法设置凸阵取样门信息
- ColorImageEngineServer 类使用 Get_D_PRFList_Convex 方法获取凸阵 PRF 信息。
- ColorImageEngineServer 类使用 SetD_PWSamplingGateParam_PA 方法设置相控阵取样门信息
- ColorImageEngineServer 类使用 Get_D_PRFList_PA 方法获取相控阵 PRF 信息。
- ColorImageEngineServer 类使用 SetSamplingLineParam_Convex 方法设置线阵取样线信息。
- ColorImageEngineServer 类使用 SetD_PWSamplingGateParam_Convex 方法设置凸阵取样线信息
- ColorImageEngineServer 类使用 SetSamplingLineParam_Phased 方法设置相控阵取样线信息
- ColorImageEngineServer 类使用 AutomaticDiameterMeasurement 方法血管经自动测量
- ColorImageEngineServer 类使用 AutomaticTGC 方法进行 TGC 自动调节
- ColorImageEngineServer 类使用 AutomaticSV 方法进行 D 模式自动取样门设置
- ColorImageEngineServer 类使用 AutoPRF 方法进行泽东 PRF 和基线调节

ColorImageEngineServer (DataSourceInterface& dsInterface);

说明

构造图像引擎，需要根据数据源接口，自定义数据源接口类，并传递进图像引擎。
SDK 中已经提供了 windows 版本的数据源接口的实现，可以直接使用。

StartImageEngine（）

说明

启动图像引擎。

在调用该方法之前需要先调用 SetCheckModeParamPath_And_ScanMode 方法设置模式和参数路径。如何是 B 模式需要调用 SetImageParam_B 方法设置 B 模式参数。如果是 C 模式需要调用 SetImageParam_B 方法和 SetImageParam_C 方法设置 B 和 C 模式参数

返回值

0：成功
-1：USB 设备开启失败
-2：处理流程启动失败
-3: 没有设置正确的 CheckModeParamPath 路径
-4: 没有设置正确的 B_ImageParam 参数
-5: 没有设置正确的 C_ImageParam 参数

StopImageEngine（）

说明
关闭图像引擎。

返回值
0：成功

bool IsRuning ()

说明
图像引擎是否在运行。

返回值
true：运行
false：停止

bool IsUSBOpen ()

说明
1.IsUSBOpen 方法当 USB 读写错误后，返回 false。不能通过此方法来判断 USB 物理断开。

2.使用系统对于 USB 设备的检测方法，可以准确判断 USB 是否是物理断开，需要用户自己实现

返回值
true： USB 设备读写正常
false：USB 设备读写不正常

void Freeze ()

说明
冻结图像，此时硬件处于非扫描状态。

返回值

void UnFreeze ()

说明

冻结图像，此时硬件处于扫描状态。

当硬件断开并重连后，此方法会重新开启 USB 设备。无需重新调用 StartImageEngine 方法

返回值

void SetUseImageHistoryCache(bool isUse, int cacheLen);

说明

设置是否使用图像引擎自带的图像历史缓存

参数

bool isUse

true：通过调用“GetImageDisplayCacheDataCount”方法获取缓存图像的数量

通过调用“GetImageDisplayCacheData（ImageDisplayData＆pImageDisplayData，int index）”获取任何一帧图像缓存

false：不使用图像缓存，“GetImageDisplayCacheDataCount”方法返回 0，

“GetImageDisplayCacheData（ImageDisplayData＆pImageDisplayData，int index）”方法返回 0

int cacheLen

如果 isUse 设置为 true，则 cacheLen 是历史缓存数据长度。cacheLen 设置范围是 100-1000。

返回值

void SetFrameDataListener(FrameDataListener* listener);

说明

设置监听器

当有硬件消息时，会通过 FrameDataListener.HardWareMsgUpdated 方法返回硬件消息。

帧监听器需要自定义实现

Native 继承 FrameDataListener 实现的监听器

int GetImageDisplayData (UImagingData &pUImagingData)

说明
获取实时显示图像。

参数
UImagingData &pUImagingData　显示数据存储对象

返回值
0：没有显示数据
1：获取到一个显示数据

pUImagingData. m_bBHadData == true ： 包含 B 图像数据
pUImagingData. m_bCHadData == true ： 包含 B&C 图像数据
pUImagingData. m_bDPWHadData == true ： 包含 DPW 图像数据
pUImagingData. m_bMHadData == true ： 包含 M 图像数据

int GetImageDisplayCacheData (UImagingData &pUImagingData, int index)

说明
获取缓存图像数据。

参数
UImagingData &pUImagingData 显示数据存储对象
int index　获取显示数据的索引，先调用 GetImageDisplayCacheDataCount 方法获取缓存的数据量 Num，索引的范围是 0 - (Num-1)。
返回值
0：没有显示数据，当 index 值超出范围时返回值也是 0
1：获取到一个显示数据

int GetImageDisplayCacheDataCount ()

说明
获取缓存图像数据数量。

返回值
返回实际缓存图像数据的数量，值范围 0-100。

int GetImageDisplayCacheData_D_PW(UImagingData *pUImagingData, int dataStartNumber,int dataLen);

说明

获取缓存的多普勒数据

参数

UImagingData* pUImagingData	多普勒数据存储对象 长度== dataLen
int dataStartNumber	多普勒数据起始编码
int dataLen	要获取的多普勒数据长度

返回值

0：没有数据返回

1：获取多普勒数据

int GetImageDisplayCacheData_BM(int OneScreenWidth, int index, UImagingData *pUImagingData,int &StartPixel, int &MIndexPixel, double MIndexAppend,bool getOneScreen, double MPosOnScreen = 1.0);

说明

获取缓存的 M 数据。

参数

int OneScreenWidth	一屏的宽度
int index	数据缓存索引，从 0 开始
UImagingData& pUImagingData	B & M 存储对象
int& StartPixel	获取的数据，在宽度为 OneScreenWidth 的屏幕上，开始绘制的位置。绘制坐标索引从 0 开始，例如：宽 480，绘制索引 0-479。
int& MIndexPixel	获取的 M 数据，在宽度为 OneScreenWidth 的屏幕上，Index 数据绘制的结束位置。绘制坐标索引从 0 开始，例如：宽 480，绘制索引 0-479。
double MIndexAppend	M 附加索引，取值范围-1.0-1.0。MIndexAppend 标识在 Index 位置上 获取前一屏到后一屏位置上的数据。默认==0。
bool getOneScreen	是否获取一屏数据 ，获取 M 图像会比实际的大一些。 当前 getOneScreen 生效时，MIndexAppend 失效。
double MPosOnScreen	在 getOneScreen==true 时，MPosOnScreen 是标识

当前 MIndex 在一屏数据的位置。默认==1。 取值范围 0.

返回值

0：没有数据返回

1：获取数据

int SetCheckModeParamPath_And_ScanMode(char *checkModeParamPath,ScanMode::ScanModeEnum scanmode)

说明

设置检查模式参数路径和扫描模式。

在 StartImageEngine 前，需要先调用 SetCheckModeParamPath_And_ScanMode 方法，SetImageParam_B 方法/SetImageParam_C 方法对图像引擎初始化。

参数

char* checkModeParamPath 检查模式参数路径，例：

C:/CIESDKUseDemo/IES_SDK/Param/L5-10/Vascular

ScanMode::ScanModeEnum scanmode 扫描模式

返回值

0：成功

-1：目录不存在

-2：目录结构不正确

-999:不支持当前设备

int SetImageParam_B(B_ImageParam b_ImageParam);

说明

设置 B 图像参数。

参数

B_ImageParam b_ImageParam B 模式参数

返回值

0：成功

-1：有参数没有正确初始化

-2：参数在接口中会进行保护，当参数设置超出范围时，将返回-2

-3: SetCheckModeParamPath_And_ScanMode 方法调用失败

int SetImageParam_C(C_ImageParam c_ImageParam);

说明

设置 C 图像参数。

参数

C_ImageParam c_ImageParam　　C 模式参数

返回值

0：成功
-1：有参数没有正确初始化
-2：参数在接口中会进行保护，当参数设置超出范围时，将返回-2
-3: SetCheckModeParamPath_And_ScanMode 方法调用失败

int SetImageParam_D_PW(D_PW_ImageParam d_PW_ImageParam);

说明

设置 DPW 图像参数。

参数

D_PW_ImageParam d_PW_ImageParam　　DPW 模式参数

返回值

0：成功
-1：有参数没有正确初始化
-2：参数在接口中会进行保护，当参数设置超出范围时，将返回-2
-3: SetCheckModeParamPath_And_ScanMode 方法调用失败

int SetImageParam_M(M_ImageParam m_ImageParam);

说明

设置 M 图像参数。

参数

M_ImageParam m_ImageParam　M 模式参数

返回值
0：成功
-1：有参数没有正确初始化
-2：参数在接口中会进行保护，当参数设置超出范围时，将返回-2
-3: SetCheckModeParamPath_And_ScanMode 方法调用失败

int SetFocusArea(float fFocusArea);

说明

设置任意焦点。

参数
float fFocusArea　设置当前焦点的值，焦点值取值范围限定在从 GetCurrentFocusArray 方法获取焦点最小值，到当前图像深度的最大值。

返回值
0：成功
-1：有参数没有正确初始化

~~int GetCurrentFocusArea(float& startDepth, float& endDepth);~~

~~**说明**~~ 该接口废弃
~~获取当前焦区的起始深度。~~
~~**参数**~~
~~float& startDepth 焦区开始深度~~
~~float& endDepth 焦区结束深度~~

~~**返回值**~~
~~0：成功~~

int GetCurrentFocusArray(float* FocusArray, int& FocusArrayLen);

说明

获取当前焦点的数值数组。
需要正确设置了 B 的参数后调用

参数
float* FocusArray　存储焦点数值的数组 单位 mm
int& FocusArrayLen 焦点数组的长度。默认长度是 8。FocusArray [0—5]是焦点 1—6 对应深度，FocusArray [6 和 7]是全域的起始深度

返回值
0：成功
-1：失败，数组未空或长度不对

int GetCurrentFocusArrayByParamPath(char* checkModeParamPath,int m_nDepth_level, int m_nHarmonic,float* FocusArray, int& FocusArrayLen);

说明
通过检查模式参数路径，来获取当前焦点的数值数组。

参数
char* checkModeParamPath 检查模式参数路径，例：
C:/CIESDKUseDemo/IES_SDK/Param/L5-10/Vascular
int m_nDepth_level　深度档位 0~n ，n 值从 ProbeInfo 类的 m_nShowDepthLevel 获取
int m_nHarmonic　是否开启谐波 0：关闭 1：开启
float* FocusArray　存储焦点数值的数组 单位 mm
int& FocusArrayLen　焦点数组的长度。默认长度是 8。FocusArray [0—5]是焦点 1—6 对应深度，FocusArray [6 和 7]是全域的起始深度

返回值
0：成功
-1：失败，数组未空或长度不对

float GetCurrentFrameRate()

说明

获取当前探头下，当前深度的帧率,参数预制值，可能比实际值高。

参数

返回值

int GetProberInfo(ProbeInfo& probeInfo, char * checkModeParamPath)

说明

获取探头信息

参数

ProbeInfo& probeInfo　探头参数信息
char * checkModeParamPath 检查模式参数路径 例：C:/CIESDKUseDemo/IES_SDK/Param/L5-10/Vascular

返回值
0：成功
-1：目录错误

~~int GetD_PWInfo(D_PWInfo &d_PWInfo, char *checkModeParamPath)~~

~~**说明**~~ 该接口废弃

~~获取 DPW 信息~~

~~**参数**~~

~~D_PWInfo &d_PWInfo　D PW 参数信息~~
~~char * checkModeParamPath 检查模式参数路径 例：C:/CIESDKUseDemo/IES_SDK/Param/L5-10/Vascular~~

~~**返回值**~~
~~0：成功~~
~~-1：目录错误~~

int GetImageWidthPixels();

说明

获取图像数据的像素宽度。

在获取宽度之前，需要调用 SetCheckModeParamPath_And_ScanMode 方法设置检查模式参数路径，

然后调用 SetImageParam_B 方法设置帧的宽度，高度和当前深度。

修改检查参数路径或修改图像的宽度，高度和当前深度时，可以调用 GetImageWidthPixels 方法以获取新的宽度。

参数

返回值

-1 : SetCheckModeParamPath_And_ScanMode 方法调用失败

-2 : 图像的宽，高，深度未设置

>0 : 图像的宽度（像素）

int GetImageHeightPixels();

说明

获取图像数据的像素宽度。

在获取高度之前，需要调用 SetCheckModeParamPath_And_ScanMode 方法设置检查模式参数路径，

然后调用 SetImageParam_B 方法设置帧的宽度，高度和当前深度。

修改检查参数路径或修改图像的宽度，高度和当前深度时，可以调用 GetImageHeightPixels 方法以获取新的高度。

参数

返回值

-1 : SetCheckModeParamPath_And_ScanMode 方法调用失败

-2 : 图像的宽，高，深度未设置

>0 : 图像的高度（像素）

int GetColorMap(unsigned int *B_MapArray, int &B_MapArrayLen, unsigned int *C_MapArray,int &C_MapArraylen);

int GetColorMap_B(unsigned int *B_MapArray, int &B_MapArrayLen);

int GetColorMap_C(unsigned int *C_MapArray, int &C_MapArraylen);

int GetColorMap_D(unsigned int *D_MapArray, int &D_MapArrayLen);

int GetColorMap_M(unsigned int *M_MapArray, int &M_MapArraylen);

说明

获取 map。

参数

unsigned int* B_MapArray	B Map	B map 绘制方向由上至下
int &B_MapArrayLen	B_MapArrayLen = 256	
unsigned int* C_MapArray	C Map	C map 绘制方向由上至下，由左至右
int &C_MapArraylen	C_MapArrayLen = 256*16	
unsigned int *D_MapArray	D Map	Dmap 绘制方向由上至下
int &D_MapArrayLen	D_MapArrayLen = 256	
unsigned int *M_MapArray	M Map	M map 绘制方向由上至下
int &M_MapArrayLen	M_MapArrayLen = 256	

返回值

0： 成功

-1：参数设置不正确

int GetHardWareInfo (HWInfo &hardWareInfo);

说明

获取当前连接设备的硬件信息

参数

返回值

0：成功获得硬件版本
-2: 发送 usb 消息失败，请检查 usb 连接
-3: 没获取到硬件信息请重置

int SetCSamplingFrameParam_Quadrangle(int Depth, double nFWidthP, double nFHeightP,double nIWidthP, double nIHeightP, double SX,double SY, double SWidthLen, double SHeightLen,int LDAngle);

说明
设置 C 四边形取样框参数
设置完取样们参数后，调用 Get_C_PRFList_Quadrangle 方法获取新得 PRF 列表，然后设置 C 的 PRFLevel 参数
帧图像左上角为原点 0,0，向右 X 轴正方向，向下 Y 轴正方向

参数

int Depth　　//当前显示的最大深度 mm
double nFWidthP　//帧图像宽度像素，SDK 中实际上传的值
double nFHeightP　//帧图像高度像素，SDK 中实际上传的值
double nIWidthP　//实际图像区域宽度像素 SDK 中实际上传的值 GetImageWidthPixels 方法获取
double nIHeightP　//实际图像区域高度像素 SDK 中实际上传的值 GetImageHeightPixels 方法获取
double SX　　//取样框左上点坐标 X，相对于实际图像区域宽度像素，如果显示有进行放大缩小操作，请根据比例计算出实际的值。
double SY　　//取样框左上点坐标 Y，相对于实际图像区域宽度像素，如果显示有进行放大缩小操作，请根据比例计算出实际的值。
double SWidhtLenP//取样框宽像素（四边形的宽），如果显示有进行放大缩小操作，请根据比例计算出实际的值。
double SHeightLenP//取样框高像素（四边形的高），如果显示有进行放大缩小操作，请根据比例计算出实际的值。
int LDAngle　//发射偏转角度 单位度

返回值

-2：发送命令失败 请检查 USB 是否连接正常
-3：多普勒参数获取失败
0 ：成功

int Get_C_PRFList_Quadrangle(int Depth, double nIHeightP, double SY, double SHeightLen, int LDAngle, double *fPrfList, int &fPrfListLen,double &fTxFrequency);

说明
获取线阵 PRF

参数

int Depth //当前显示的最大深度 mm
double nIHeightP //实际图像区域高度像素 SDK 中实际上传的值。 GetImageHeightPixels 方法获取
double SY //取样框左上点坐标 Y，相对于帧图像的位置，如果显示有进行放大缩小操作，请根据比例计算出实际的值。
double SHeightLenP //取样框高像素（四边形的高），如果显示有进行放大缩小操作，请根据比例计算出实际的值。
int LDAngle //发射偏转角度
double* fPrfList //PRF 列表
int &fPrfListLen // == 8
double& fTxFrequency//发射频率，当前 prf 列表对应的频率

返回值

-1 : fPrfListLen 长度小于 8
-2：没有获取到当前模式下 CPRF 参数
-3：根据当前参数计算的取样框深度超出参数范围
-4：超出 CPRF 参数范围
0 ：成功

int SetCSamplingFrameParam_Sector(int Depth, double nFWidthP, double nFHeightP, double nIWidthP,double nIHeightP, double SCToPCCP, double SHeight,double SectorAngle, double SectorCenterAngle);

说明
设置 C 扇形形取样框参数
设置完取样们参数后，调用 Get_C_PRFList_Sector 方法获取新得 PRF 列表，然后设置 C 得 PRFLevel 参数
帧图像左上角为原点 0,0，向右 X 轴正方向，向下 Y 轴正方向

参数
int Depth //当前显示的最大深度 mm
double nFWidthP //帧图像宽度像素，SDK 中实际上传的值
double nFHeightP //帧图像高度像素，SDK 中实际上传的值
double nIWidthP //实际图像区域宽度像素 SDK 中实际上传的值 GetImageWidthPixels 方法获取
double nIHeightP //实际图像区域高度像素 SDK 中实际上传的值 GetImageHeightPixels 方法获取
double SCToPCCP //扇形取样框的中心点到探头圆心的长度，单位像素 如果显示有进行放大缩小操作，请根据比例计算出实际的值。
double SHeight //扇形取样框的高度，上下弧的差，单位像素 如果显示有进行放大缩小操作，请根据比例计算出实际的值。
double SectorAngle//扇形取样框的夹角 单位度
double SectorCenterAngle//扇形取样框的中心点到探头圆心的连线与 X 轴正方向的夹角，顺时针方向，正直 单位度

返回值
-2：发送命令失败 请检查 USB 是否连接正常
-3：多普勒参数获取失败
0 ：成功

int Get_C_PRFList_Sector(int Depth, double nIHeightP, double SCToPCCP, double SHeight,double *fPrfList, int &fPrfListLen, double &fTxFrequency);

说明

获取凸阵 PRF

参数

int Depth //当前显示的最大深度 mm

double nIHeightP //实际图像区域高度像素 SDK 中实际上传的值。 GetImageHeightPixels 方法获取

double SCToPCCP //扇形取样框的中心点到探头圆心的长度，单位像素 如果显示有进行放大缩小操作，请根据比例计算出实际的值。

double SHeight //扇形取样框的高度，上下弧的差，单位像素 如果显示有进行放大缩小操作，请根据比例计算出实际的值。

double* fPrfList //PRF 列表

int &fPrfListLen // == 8

double& fTxFrequency//发射频率，当前 prf 列表对应的频率

返回值

-1 : fPrfListLen 长度小于 8

-2 : 没有获取到当前模式下 CPRF 参数

-3 : 根据当前参数计算的取样框深度超出参数范围

-4 : 超出 CPRF 参数范围

0 : 成功

int SetD_PWSamplingGateParam_Line(float SX, float SY, float depth, float fIWidthP, float fIHeightP, int launchDeflectionAngle,int samplingVolume);

说明

设置 线阵 D_PW 取样门参数

参数

float SX	取样门中心点 Y 值
float SY	取样门中心点 Y 值
float depth	当前深度 mm
float fIWidthP	当前 B 显示宽度像素
float fIHeightP	当前 B 显示像素
int launchDeflectionAngle	发射偏转角度
int samplingVolume	取样门宽度 1-10mm

返回值

-2： 发送命令失败 请检查 USB 是否连接正常
-3： 多普勒参数获取失败
0 ： 成功

int Get_D_PRFList_Line(float SY, float depth, float fIHeightP, int launchDeflectionAngle,double *fPrfList, int &fPrfListLen, double &fTxFrequency);

说明

获取线阵 PRF

参数

float SY	取样门中心点 Y 值
float depth	当前深度,mm
float fIHeightP	当前 B 显示像素
int launchDeflectionAngle	发射偏转角度
double* fPrfList	PRF 列表
int &fPrfListLen	== 8
double& fTxFrequency	发射频率，当前 prf 列表对应的频率

返回值

-1 : fPrfListLen 长度小于 8
-2： 没有获取到当前模式下 CPRF 参数
-3： 根据当前参数计算的取样框深度超出参数范围

-4：超出 CPRF 参数范围
0 ：成功

int SetD_PWSamplingGateParam_Convex(float SX, float SY, float depth, float fIWidthP,float fIHeightP, float SectorCenterAngle,int samplingVolume);

说明
设置 凸阵 D_PW 取样门参数
参数

float SX	取样门中心点 Y 值
float SY	取样门中心点 Y 值
float depth	当前深度 mm
float fIWidthP	当前 B 显示宽度像素
float fIHeightP	当前 B 显示像素
float SectorCenterAngle	扇形取样框的中心点到探头圆心的连线与 X 轴正方向的夹角，顺时针方向，正直 单位度
int samplingVolume	取样门宽度 1-10mm

返回值
-2：发送命令失败 请检查 USB 是否连接正常
-3：多普勒参数获取失败
0 ：成功

int Get_D_PRFList_Convex(float SX, float SY, float depth, float fIWidthP, float fIHeightP, double *fPrfList, int &fPrfListLen, double &fTxFrequency);

说明
获取 凸阵 PRF
参数

float SX	取样门中心点 Y 值
float SY	取样门中心点 Y 值
float depth	当前深度 mm
float fIWidthP	当前 B 显示宽度像素
float fIHeightP	当前 B 显示像素
double* fPrfList	PRF 列表

int &fPrfListLen == 8
double& fTxFrequency 发射频率，当前 prf 列表对应的频率

返回值

-1 : fPrfListLen 长度小于 8
-2 : 没有获取到当前模式下 CPRF 参数
-3 : 根据当前参数计算的取样框深度超出参数范围
-4 : 超出 CPRF 参数范围
0 : 成功

int SetD_PWSamplingGateParam_PA(float SX, float SY, float depth, float fIWidthP, float fIHeightP, float SectorCenterAngle,int samplingVolume);

说明

设置 相控阵 D_PW 取样门参数

参数

float SX 取样门中心点 Y 值
float SY 取样门中心点 Y 值
float depth 当前深度 mm
float fIWidthP 当前 B 显示宽度像素
float fIHeightP 当前 B 显示像素
float SectorCenterAngle 扇形取样框的中心点到探头圆心的连线与 X 轴正方向的夹角，顺时针方向，正直 单位度
int samplingVolume 取样门宽度 1-10mm

返回值

-2 : 发送命令失败 请检查 USB 是否连接正常
-3 : 多普勒参数获取失败
0 : 成功

int Get_D_PRFList_PA(float SX, float SY, float depth, float fIWidthP, float fIHeightP, double *fPrfList, int &fPrfListLen, double &fTxFrequency);

说明

获取 相控阵 PRF

参数

float SX	取样门中心点 Y 值
float SY	取样门中心点 Y 值
float depth	当前深度 mm
float flWidthP	当前 B 显示宽度像素
float flHeightP	当前 B 显示像素
double* fPrfList	PRF 列表
int &fPrfListLen	== 8
double& fTxFrequency	发射频率，当前 prf 列表对应的频率

返回值

-1 : fPrfListLen 长度小于 8
-2 : 没有获取到当前模式下 CPRF 参数
-3 : 根据当前参数计算的取样框深度超出参数范围
-4 : 超出 CPRF 参数范围
0 ： 成功

int SetSamplingLineParam_Line(float SLX, float depth, float flWidthP, float flHeightP);

说明

设置 M 取样线参数 线阵

参数

float SLX	取样线坐标 X 值
float depth	当前深度 mm
float flWidthP	当前 B 显示宽度像素
float flHeightP	当前 B 显示像素

返回值

0 ： 成功
-2 ： 失败

int SetSamplingLineParam_Convex(float SectorCenterAngle, float depth, float flWidthP,float flHeightP);

说明

设置 M 取样线参数 凸阵

参数

float SectorCenterAngle　　取样线的与 X 轴正方向的夹角，顺时针方向，正直 单位度
float depth　　当前深度 mm
float flWidthP　　当前 B 显示宽度像素
float flHeightP　　当前 B 显示像素

返回值

0　：成功
-2　：失败

int SetSamplingLineParam_Phased(float SectorCenterAngle, float depth, float flWidthP,float flHeightP);

说明

设置 M 取样线参数 相控阵

参数

float SectorCenterAngle　　取样线的与 X 轴正方向的夹角，顺时针方向，正直 单位度
float depth　　当前深度 mm
float flWidthP　　当前 B 显示宽度像素
float flHeightP　　当前 B 显示像素

返回值

0　：成功
-2　：失败

int AutomaticDiameterMeasurement(int index, float PointOnBImageX,float PointOnBImageY,float &CenterOfVascularDiameterX, float &CenterOfVascularDiameterY, float &VascularRadius);

说明

根据 B 图像计算血管半径，冻结后调用。

参数

int index　　获取显示数据的索引，先调用 GetImageDisplayCacheDataCount 方法获取缓存的数据量 Num，索引的范围是 0 - (Num-1)。
float PointOnBImageX　B 图像上选取的一点的 X 轴坐标
float PointOnBImageY　B 图像上选取的一点的 Y 轴坐标
float &CenterOfVascularDiameterX　　返回血管圆心的 X 轴坐标

float &CenterOfVascularDiameterY	返回血管圆心的 Y 轴坐标
float &VascularRadius	返回血管半径

返回值

0 ：成功

-1 ：失败

int AutomaticTGC(float *TGCArray, int &TGCArrayLne);

说明

根据图像自定计算合适的 TGC 参数，B 或 C 模式非冻结时使用。

参数

float SectorCenterAngle	取样线的与 X 轴正方向的夹角，顺时针方向，正直 单位度
float depth	当前深度 mm
float flWidthP	当前 B 显示宽度像素
float flHeightP	当前 B 显示像素

返回值

0 ：成功

-1 ：失败

int AutomaticSV(float& svx, float& svy, float& svH, int& svAngleLevel);

说明

在 D 模式，C 模式激活时调用，根据 C 模式图像自动计算出 D 的取样门合适的位置。

使用线阵探头时使用，一定取样窗后 1s 后可以调用此方法。

参数

float& svx	取样门 X 坐标 在图像区域
float& svy	取样门 Y 坐标 在图像区域
float& svH	取样容积高度像素
int& svAngleLevel	取样门偏转角度

返回值

0 ：成功

-1 ：失败

int AutoPRF(int& PRF_level, int& Baseline_level);

说明

根据 D 图像自动计算 PRF 和基线。进入 D 模式 1.5s 后可以调用此方法。

参数

int& PRF_level　　PRF 档位

int& Baseline_level 基线档位

返回值

0　：成功

-2　：缓存数据不够

-3　：缓存数据不够

int GetNewColorMap_B(int ctrlPoint_X[256], int ctrlPoint_Y[256], int useCtrlPointLen, unsigned int* B_MapArray, int& B_MapArrayLen);

说明

输入 B 模式灰阶的控制点，输出 256 阶灰阶 Map。

参数

int ctrlPoint_X[256]　　控制点 X 轴坐标（取值范围 0-255）

int ctrlPoint_Y[256]　　控制点 Y 轴坐标（取值范围 0-255）

int useCtrlPointLen,

unsigned int* B_MapArray　返回灰阶 Map。该值只有低 8 未有效。

例：B_MapArray[1] = 0x000000FF。

在上位机实际使用时，需要重新整理成 RGB 值。例：0x000000FF →0x00FFFFFF。

int& B_MapArrayLen　　灰阶 Map 的长度。传入参数正确，返回值==256。

返回值

0　：成功

-1　：B_MapArrayLen 长度 < 256

-2　：useCtrlPointLen　长度 > 256

int SetColorMap_B(unsigned int* B_MapArray, int B_MapArrayLen);

说明

输入 B 模式第 9 档灰阶值。利用 GetNewColorMap_B 方法获取的灰阶值，转换成 RGB 值后，

调用此方法，将第 9 档灰阶值传入到 SDK 中。
也可传入自定义灰阶值。

参数

unsigned int* B_MapArray　设置的灰阶 Map。。
int& B_MapArrayLen　　灰阶 Map 的长度。长度需要==256。

返回值

0　：成功
-1　：B_MapArrayLen 长度 != 256

ImageDataInfo Class

头文件

ShowFrame.h

说明

ImageDataInfo 类存储图像信息

```
int m_nFrameNumber;            //帧号
int m_nDepth;                  //当前深度

int m_nFrameWidthPixels;       //帧横向分辨率
int m_nFrameHeightPixels;      //帧纵向分辨率
int m_nImageWidthPixels;       //图像横向分辨率
int m_nImageHeightPixels;      //图像纵向分辨率

int m_nHour;                   //时
int m_nMinute;                 //分
int m_nSecond;                 //秒
int m_nMillisecond;             //毫秒

float m_fFrameRate;            //帧率
float m_fResolution;           //分辨率 单位 像素/毫米。

int C_EmitStartLineNo;         //C模式 发射起始线
int C_EmitStopLineNo;          //C 模式 发射结束线
```

EnvelopeInfo Class

头文件

ShowFrame.h

说明

EnvelopeInfo 类存储 PW 包络线数据

```
bool bHadData;//是否有数据
short nEnvelopeMax; //最大包络
short nEnvelopeMean;//平均包络
```

包络线数据只有在回放时才会计算，实时显示不计算该数据

D_PWDataInfo Class

头文件

ShowFrame.h

说明

PWDataInfo 类存储 PW 包络线数据和当前数据编号

```
EnvelopeInfo envelopeInfo;              //包络线
int m_nDateNum;                         //数据编号 当收到数据变化是 0 时，界面清屏

float m_AutoMeasureAngle;               //默认值是-999不生效的值 自动测量角度 -90 ~ 90
float m_BloodRadius_T;                  //默认值是-999不生效的值 像素
float m_BloodRadius_B;                  //默认值是-999 不生效的值 像素
```

MDataInfo Class

头文件

ShowFrame.h

说明

MDataInfo 类存储 M 线数据和

```
int m_nFrameNumber;             //帧号
int m_nImageHeightPixels;       //图像纵向分辨率
int m_nMLineNum;                //M谱线数量 当前帧数据中包含M谱线的数量
bool m_bClearScreen;            //清屏
```

ShowFrame Class

头文件

ShowFrame.h

说明

ShowFrame 类存储图像数据和图像信息

本类可以获取 B/C/D/M 的数据，及 B/C/D/M 的图像信息

UImagingData Class

头文件

ShowFrame.h

说明

UImagingData 类存储超声图像数据和图像信息

```
ShowFrame m_B_Data;        //B 数据
ShowFrame m_C_Data;        //C 数据 包含B和C的显示数据
ShowFrame m_D_PW_Data;     //D PW 数据 包含D_pw数据
ShowFrame m_M_Data;        //M 数据

bool m_bBHadData;      //包含B数据
bool m_bCHadData;      //包含 C 数据
bool m_bDPWHadData;    //包含DPW数据
bool m_bMHadData;      //包含 M 数据
```

当 bBHadData == true && bCHadData == true 时，此时是 B&C 双实时状态。
通过设置 C_ImageParam. m_bUse_B_BC_Mode ==true 来实现切换到 B&C 双实时状态
此时 B 图像和 C 图像的宽高相同

当 bBHadData == true && bMHadData == true 时，此时是 B&双实时状态。

Version Class

头文件

CIESParamter.h

说明

SDK 版本类

```
int Major;        //主版本
int Minor;        //次版本
int Revision;     //修订号
```

ScanMode Class

头文件

CIESParamter.h

说明

ScanMode 类标识当前探头需要设置的扫描模式。

```
enum ScanModeEnum        //扫描模式
{
    B = 0x01,            //B 模式
    C = 0x02,            //C 模式
    D_PW = 0x04,         //D_PW 模式
    D = 0x10,            //D 模式
    BC = B | C,          //B&C 模式
    M = 0x20,            //M 模式
    BM = B | M           //B&M 模式
};

static bool ContainBScanMode(ScanModeEnum scan_mode)    //包含B扫描模式

static bool ContainCScanMode(ScanModeEnum scan_mode)    //包含C扫描模式

static bool ContainBCScanMode(ScanModeEnum scan_mode)    //包含B&C扫描模式

static bool ContainDPWScanMode(ScanModeEnum scan_mode)    //包含D_PW扫描模式

static bool ContainDScanMode(ScanModeEnum scan_mode)    //包含D扫描模式
```

```
static bool ContainMScanMode(ScanModeEnum scan_mode)     //包含M扫描模式

static bool ContainBMScanMode(ScanModeEnum scan_mode)     //包含B&M扫描模式
```

B_ImageParam Class

头文件

CIESParamter.h

说明

B_ImageParam 类存储用来下发到图像引擎的 B 参数。

```
enum B_ImageParamType {
    All_Param = 0x7FFF,                         //所有参数
    Width_Param = 0x0001,                       //DSC宽（像素） 输出图像的宽
    Height_Param = 0x0002,                      //DSC高（像素）  输出图像的高
    Depth_level_Param = 0x0004,                 //深度档位
    Gain_Param = 0x0008,                        //增益
    DR_Param = 0x0010,                          //动态范围
    SRI_level_Param = 0x0020,                   //图像增强档位
    Correlation_level_Param = 0x0040,           //帧相关档位
    TGC_Param = 0x0080,                         //六段TGC
    GrayColorMap_level_Param = 0x0100,          //灰度映射Map档位
    PseudoColorMap_level_Param = 0x0200,        //伪彩映射Map档位
    Harmonic_level_Param = 0x0400,              //谐波
    Frequency_level_Param = 0x0800,             //频率
    FocusArea_level_Param = 0x1000,             //焦区
    NE_Param = 0x2000,                          //针增强
    NE_Theta_Param = 0x4000,                    //针增强角度
};

int m_B_ImageParamType;             //参数类型 B_ImageParamType 变量的集合

int m_nWidth;                   //DSC宽（像素）输出图像的宽 （0~2048）
int m_nHeight;                  //DSC高（像素）输出图像的高  （0~2048）
int m_nDepth_level;             //深度档位 （0~n，n一般为6档）
float m_fGain;                  //增益范围 （1~100）
float m_fDR;                    //动态范围   1~100）
int m_SRI_Level;                //图像增强档位 （1~8）
int m_nCorrelation_Level;       //帧相关系数 （1~4）
float m_fTGC[6];                //六段TGC （-15~15）
int m_nGrayColorMap_level;      //灰度映射Map档位 （1~9，9为自定义档位）
int m_nPseudoColorMap_level;    //伪彩映射Map档位 （0~8，0为关）
int m_nHarmonic;                //谐波    （0:基波；1：谐波）
int m_nFrequency;               //频率    （0：低频;1：中频；2：高频；）
int m_nFocusArea;               //焦区    （0：焦点 1;1：焦点 2；2：焦点 3；3：焦点 4；4：焦点 5；5：焦点 6；6：）
int m_nNE;                      //针增强 0:关;1:开;
```

```
int m_nNE_Theta;            //针增强角度: -30° - 30° ;（角度值从
ProbeInfo.m_pB_NE_Theta_angle 参数中获取）
```

C_ImageParam Class

头文件

CIESParamter.h

说明

C_ImageParam 类存储用来下发到图像引擎的 C 参数。

```
enum C_ImageParamType {
    All_Param = 0x07FF,                         //所有参数
    Gain_Param = 0x0001,                    //增益
    WallFilter_level_Param = 0x0002,        //壁滤波档位
    ColorPriority_level_Param = 0x0004,     //彩色优先度档位
    FrameCorrelation_level_Param = 0x0008,  //彩色帧相关档位
    Color_Mode_Param = 0x0010,              //模式
    ColorMap_level_Param = 0x0020,          //映射Map档位
    ColorMap_Inversion_Param = 0x0040,      //图像翻转
    PRF_Level_Param = 0x0080,               //PRF档位
    Theta_Param = 0x0100,                   //发射偏转角度
    B_BC_Mode_Param = 0x0200,               //使用 B & BC模式
    Speed_Param = 0x0400,                   //速度
};

int m_C_ImageParamType;             //参数类型 C_ImageParamType 变量的集合

float m_fGain;                      //增益 (0~100)
int m_nWallFilter_level;        //壁滤波档位 (0~4)
int m_nColorPriority_level;         //彩色优先度档位（0~4）
int m_nFrameCorrelation_level; //彩色帧相关档位（0~4)
int m_nColor_mode;                  //模式（0：速度模式；1：能量模式；）
int m_nColorMap_level;          //映射Map档位 （0~11)
int m_nColorMap_inversion;      //图像翻转 （0：正常；1：翻转）
int m_nPRF_Level;               //PRF档位 （0~7)
float m_fTheta;                     //发射偏转角度 （角度值从ProbeInfo.m_pC_Tx_angle参数
中获取）
bool m_bUse_B_BC_Mode;          //使用 B & BC模式
int m_nSpeed;                   //速度 （0：低速，1：高速）
```

D_PW_ImageParam Class

头文件

CIESParamter.h

说明

D_PW_ImageParam 类存储用来下发到图像引擎的 D PW 参数

```
class  D_PW_ImageParam
{
    public:
        D_PW_ImageParam();
        ~D_PW_ImageParam();
        D_PW_ImageParam &operator=(const D_PW_ImageParam &s);
        enum D_PW_ImageParamType {
            All_Param = 0x3FFF,
            Width_Param = 0x0001,
            Height_Param = 0x0002,
            PRF_Rate_Param = 0x0004,
            Gain_Param = 0x0008,
            DR_Param = 0x0010,
            Frequency_Param = 0x0020,
            Time_Param = 0x0040,
            BaseLine_Level_Param = 0x0080,
            Wall_Level_Param = 0x0100,
            Sampling_Volume_Param = 0x0200,
            Inversion_Param = 0x0400,
            Speed_Param = 0x0800,
            GrayColorMap_level_Param = 0x1000,
            PseudoColorMap_level_Param = 0x2000
        };

        int m_D_PW_ImageParamType;     //参数类型

        int m_nWidth;                  //显示像素宽度   最大值2048 需要是32的整数倍
        int m_nHeight;                 //显示像素高度 最大值2048 需要是32的整数倍
        int m_prf_rate;                //PRF rate
        float m_fGain;                 //增益范围 -20- 20 默认0
        float m_fDR;                   //动态范围    20 - 60 默认40
        float m_fFrequency;            //频率
        float m_fTime;                 //显示时间
        int m_nBaseLineLevel;          //基线level -3 - 3
        int m_nWall_level;             //壁滤波Level 0 - 2
        int m_nSamplingVolume;         //取样门宽度 1- 10mm
        int m_nInversion;              //图像翻转 正常：0;   翻转：1
```

```
        int m_nSpeed;                 //速度 0：低速，1：高速
        int m_nGrayColorMap_level;    //灰度映射Map档位 1 - 8
        int m_nPseudoColorMap_level;  //伪彩映射Map档位，0为默认 0 - 8
    };
```

M_ImageParam Class

头文件

CIESParamter.h

说明

M_ImageParam 类存储用来下发到图像引擎的 M 参数

```
class  M_ImageParam
{
    public:
        M_ImageParam();
        ~M_ImageParam();
        M_ImageParam &operator=(const M_ImageParam &s);
        enum M_ImageParamType {
            All_Param = 0x003F,
            Gain_Param = 0x0001,
            Time_Param = 0x0002,
            Width_Param = 0x0004,
            Height_Param = 0x0008,
            GrayColorMap_level_Param = 0x0010,
            PseudoColorMap_level_Param = 0x0020
        };

        int m_M_ImageParamType;       //参数类型
        float m_fGain;                //增益范围 (0-100)
        float m_fTime;                //显示时间(4s、6s、8s)
        int m_nWidth;                 //M宽（像素） Max 2048
        int m_nHeight;                //M高（像素）  Max 2048
        int m_nGrayColorMap_level;    //灰度映射Map档位 (1 - 8)
        int m_nPseudoColorMap_level;  //伪彩映射Map档位，(0 - 8)0为默认
    };
```

ProbeInfo Class

头文件

CIESParamter.h

说明

ProbeInfo 类存储的是探头的相关参数。

```
float m_fProbe_pitch;          //振元间距 mm
float m_fProbe_r;              //曲率半径 mm
float m_fProbe_lens;           //声透镜厚度 mm
int m_nProbe_element;          //振元数量
int m_nProbe_type;             //探头类型 （0：线阵；1：凸阵；2：相控阵；）
int m_nAD_downsample;          //AD降采样率 （1：1倍降采样（约40MHz），2：2倍降采样（约20MHz））
float m_fClock_frequency;      //时钟频率 （默认156.25MHz）
int m_nCH_physical_RX;        //接收通道数 ：32，16，8
int m_nADC_type;               //ADC芯片型号
int m_nRate_control;           //控制事件rate
float m_fImageAngle;           //相控阵张角
float m_fSound_velocity[4];    //声速 单位m/s，依次为常规、液体、脂肪、肌肉中的超声传播速度
int m_nHV_SW;                  //高压开关控制。
int m_nCH_physical_TX;         //发射通道数 ：64，32，16
bool m_bNE_On_Off;             //当前探头是否支持针增强 0：不支持，1：支持
bool m_bB_Tx_deflectionflag;   //B是否有发射偏转  未使用
int m_pB_Tx_angle[5];          //B发射偏转角度   未使用
标识当前探头是否支持发射偏转，和发射偏转的5个角度
bool m_bC_Tx_deflectionflag;   //C是否有发射偏转
int m_pC_Tx_angle[5];          //C发射偏转角度
标识当前探头是否支持发射偏转，和发射偏转的5个角度
bool m_bD_Tx_deflectionflag;   //D是否有发射偏转
int m_pD_Tx_angle[5];          //D发射偏转角度
标识当前探头是否支持帧增强，和增强的4个角度
bool m_bB_NE_deflectionflag;   //B是否有针增强
int m_pB_NE_Theta_angle[4];    //B针增强角度

float m_fB_fund_Tx_freq[3];    /发射显示频率：基波发射频率依次为：高，中，低，单位MHz
float m_fB_harm_Tx_freq[3];    //发射显示频率：谐波发射频率依次为：高，中，低，单位MHz
float m_fC_Tx_freq[2];         //发射显示频率 依次为高频（低速），低频（高速）
float m_fD_Tx_freq[2];         //发射显示频率 依次为高频（低速），低频（高速）

int m_nShowDepthLevel;         //显示深度档位
int *m_pDepthList;             //深度列表 单位 mm 显示需要*10
```

标识当前探头界面显示的深度列表，前台获取深度列表后，根据列表中的深度对界面进行操作。

根据探头参数计算探头的宽度

```
if(m_nProbe_type == 0) //线阵
double _proberWidth = m_fProbe_pitch * m_nProbe_element;

if(m_nProbe_type == 0) //凸阵
_proberWidth = m_fProbe_r * Math.Sin((m_fProbe_pitch / m_fProbe_r * m_nProbe_element) / 2) * 2;
```

~~D_PWInfo Class~~

~~头文件~~

~~CIESParamter.h~~

~~说明~~

~~D_PWInfo 类存储的是 DPW 相关参数。~~该参数已废弃

```
    class D_PWInfo
    {
    public:
        D_PWInfo();

        ~D_PWInfo();

        D_PWInfo &operator=(const D_PWInfo &s);

        float m_nFrequency;          //频率
        float m_fPRF[8];             // 对应深度档位
        bool m_bUseLaunchDeflection;//探头是否支持发射偏转 凸阵不支持发射偏转
        float m_nLaunchDeflectionAngle;//发射偏转角度（°） 单位度
    };
```

HWInfo Class

头文件

CIESParamter.h

说明

HWInfo 类存储硬件信息。

```
int nLogicVersion[2];                //逻辑版本号 ;m_unLogicVersion[0].m_unLogicVersion[1]
int nHWVersion[2];                   //硬件版本号 ;m_unHWVersion[0].m_unHWVersion[1]
unsigned int unLogicCompileVersion;  //逻辑编译号
unsigned short DNA1;                 //FPGA_DNA 0-15位
unsigned short DNA2;                 //FPGA_DNA 16-31位
unsigned short DNA3;                 //FPGA_DNA 32-47位
unsigned short DNA4;                 //FPGA_DNA 48-56位
float fUSBSupplyVoltage;             //USB供电电压          单位V
float fTotalCurrent;                 //总电流               单位A
float fTemperature;                  //温度                 单位 ℃
float fEmissionVoltage;              //发射电压             单位V
bool getInfoFlag;                    //获取到信息标识
int OtherMsg1;
bool ProbeCanreplaced;               //探头可更换
int IsMultiProbe;                    //0 不是多探头版本 1 两个探头版本
int unProbeID;                       //探头ID/探头插座 A ID  使用标识 0x5555探头断开
int unProbeAppendInfo;               //探头附加消息  低16位有效，0位：是否是可插拔探头；1位：是否支持按键
bool unProbeConnect;                 //探头ID/探头插座 A 0 不在位 1 在位
int unProbeID_B;                     //探头插座B ID 使用标识 0x5555探头断开
int unProbeAppendInfo_B;             //探头插座B 附加消息  低16位有效，0位：是否是可插拔探头；1位：是否支持按键
bool unProbeConnect_B;               //探头插座 B 0 不在位 1 在位
```

平板超声： unProbeID 是探头的 ID。
USB 超声： 从 ProductInfo 类中获取探头 ID。

ProductInfo Class

头文件

CIESParamter.h

说明

ProductInfo 类存储产品信息。

```
char productCategory[2];        //产品类别
short productVersion;           //产品版本
short probeID;                  //探头ID //针对USB 探头有效
char describe[25];              //描述信息
bool getInfoFlag;               //获取到信息标识
```

SNInfo Class

头文件
CIESParamter.h

说明
SNInfo 类存储设备 SN 信息

```
char serialNumber[32];                  //序列号
```

OtherInfo Class

头文件
CIESParamter.h

说明
OtherInfo 类存储设备其他信息

PCB_PCBA_Info Class

头文件
CIESParamter.h

说明
PCB_PCBA_Info 类存储硬件 PCB 版本和 PCBA 版本信息

USDeviceInfo Class

头文件
CIESParamter.h

说明
USDeviceInfo 类存储超声设备所有硬件信息

```
bool isPowerOff;            //是否处于低功耗
int VID;                    //USB VID
int PID;                    //USB PID
char DevicPath[256];        //设备唯一识别路径
int fwVersion_major;        //usb固件版主本
int fwVersion_minor;        //usb固件版子本
HWInfo hwInfo;              //硬件信息
ProductInfo productInfo;    //产品信息
SNInfo snInfo;              //SN信息
OtherInfo otherInfo;        //其他信息
PCB_PCBA_Info pcb_PCBA_Info;//PCB PCBA 信息
int usHardWareIndex;        //超声设备的索引  如何出现设备拔出和插入，请重新获取。
```

DataSourceInterface

头文件

DataSourceInterface.h

说明

DataSourceInterface 类数据源接口类。

内部已经实现，用户不需要实现，直接使用 DataSource 类即可。

```
//打开设备
virtual int OpenDevice() = 0;
//关闭设备
virtual int CloseDevice() = 0;
//设备是否开启
virtual bool IsDeviceOpen() = 0;
//写命令消息
virtual int WriteCmdMsgSync(unsigned char* pCmdMsg, int nCmdMsgLen) = 0;
//读取命令消息
virtual int ReadCmdMsgSync(unsigned char* pCmdMsg, int& nCmdMsgLen) = 0;
//读取数据 异步
virtual int ReadDataMsgAsync(unsigned char* pDataMsg, int& nDataMsgLen, int& index) = 0;
//设置读取缓存长度
virtual int SetReadDataBufferSize(int nSize) = 0;
// 读取硬件中断消息
```

```
virtual int ReadInterruptMsgSync(unsigned char* pInterruptMsg, int& nMsgLen) = 0;
//写喂狗消息
virtual int WriteFeedigDogSync() = 0;
// 获取超声设备信息 最多获取 5 个设备信息
virtual int GetUSDeviceInfo(void* usHareWareInfoArray, int& nArrayLen) = 0;
//获取超声设备信息，通过 DevicePath。
virtual int GetUSDeviceInfoByDevicePath(void* usHareWareInfo, char* DevicePath) = 0;
//硬件退出低功耗，发送命令后延迟 500ms 后，需要重新调用后台接口设置参数。
virtual int PowerOn(char* DevicePath) = 0;
//硬件进入低功耗
virtual int PowerOff(char* DevicePath) = 0;
```

```
const unsigned short USB_VID = 0x04B4;                          //USB VID
const unsigned short USB_PID1 = 0x00F0;                         //USB PID   平板超声 PID
const unsigned short USB_PID2 = 0x1003;                         //USB PID    USB 探头超声 PID
const char* CYUSB_GUID = "AE18AA60-7F6A-11d4-97DD-00010229B959"; //USB GUID
```

DataSource

头文件
DataSource.h

说明
该类是 DataSourceInterface 的实现。

FrameDataListener Class

头文件
FrameDataListener.h

说明
FrameDataListener 监听接口类。

```
//硬件消息上传
virtual void HardWareMsgUpdated(int msgType, int nValue) = 0;
```

msgType 消息类型 ： 2：探头ID消息
nValue 消息值 ：探头 ID、 0x5555（探头脱落）、0xAAAA(一线器件损坏或读取一线器件失败)

说明：探头插拔消息通过下面的新接口上传

//探头插拔消息上传
virtual void HardWareMsgSUpdated(int msgType, int* nValue, int nValueLen) = 0;
msgType 消息类型 ： 2：探头ID消息 只有一个探头座 4：探头ID消息 有两个探头座
int* nValue消息值 ：如下所示
int nValueLen 消息长度

A探头座
nValue[0] 探头ID
0x0000 (ID未烧录)、 0x5555（探头脱落）、0xAAAA(探头异常，需要重新插拔)
nValue[1] 探头附加信息
数据占16位[15:0]： [0]位代表可插拔探头(0b01); [1]位代表探头带按键(0b10);
nValue[2] 探头在位信息
0：探头未连接；1：探头连接；

B探头座
nValue[3] 探头ID
0x0000 (ID未烧录)、 0x5555（探头脱落）、0xAAAA(探头异常，需要重新插拔)
nValue[4] 探头附加信息
数据占16位[15:0]： [0]位代表可插拔探头(0b01); [1]位代表探头带按键(0b10);
nValue[5] 探头在位信息
0：探头未连接；1：探头连接；

Native Class

头文件
Native.h

说明
该类是 FrameDataListener 的实现。