模块 SDK

使用手册

V4.4.41

版本更新记录

版本	日期	修改者	更新摘要
更早版本	…	xrjiang、zhangyan	需查阅更早的 SDK 说明文档；
4.4.30	11/18/2021	xrjiang	TofDevInitParam 增加新字段：bWeakAuthority； 增加新接口：TOFD_OpenDevice_WithFd； 增加新模块类型：TOF_DEV_SEEKER07C、 Add TOF_DEV_SEEKER08A；
4.4.31	12/14/2021	xrjiang	增加新模块类型：TOF_DEV_DEMO_UPG；
4.4.32	01/05/2022	xrjiang	TofFrameData 增加新字段：pDepthData、 pDepthDataFilter； 增加新的数据结构：TofFrameDataPixelOffset
4.4.33	01/10/2022	xrjiang	TofDevInitParam 增加新字段： bDisablePixelOffset； 将数据结构 DepthCalRoi 作为公共数据类型；
4.4.34	01/19/2022	xrjiang	增加新模块类型：TOF_DEV_CLEANER01G1、 TOF_DEV_DEMO_C00P01A_NET、 TOF_DEV_LOGITECH_C525、 TOF_DEV_CHROMEBOOK；
4.4.35	02/14/2022	xrjiang	增加新模块类型：TOF_DEV_CLEANER01X；
4.4.36	02/21/2022	xrjiang	增加新的参数接口： TOF_DEV_PARAM_SensorStatusCtrl
4.4.37	03/14/2022	xrjiang	增加新的滤波类型：TOF_FILTER_RadialFusion
4.4.38	03/25/2022	xrjiang	增加新模块类型：TOF_DEV_CLEANER01F1；
4.4.39	04/01/2022	xrjiang	将部分数据结构作为公共数据类型； 增加新模块类型：TOF_DEV_MARS01H；
4.4.40	04/18/2022	xrjiang	支持将 debug 信息保存到自定义的文件中；
4.4.41	05/16/2022	xrjiang	补充 TOF_MODE 的枚举类型； 更正 RgbDData 格式

目录

1 概述

1.1 介绍

舜宇 TOF 模块是机器视觉的 3D Camera 产品, 它主要的功能:

1. 实时 3D 位置信息测量.
2. 实时输出 IR 图像数据.
3. 支持 AE 曝光和手动曝光两种曝光模式（部分 TOF 模组和 TOF 模块支持此功能）
4. 支持 HDRZ 功能（部分 TOF 模组和 TOF 模块支持此功能）
5. 支持多种滤波功能（不同的 TOF 模组和 TOF 模块支持的滤波功能不相同）
6. 支持多种的模式（不同的 TOF 模组和 TOF 模块支持的模式不相同）
7. 支持 RGB 图像实时采集（部分 TOF 模块具有此功能）
8. 支持 RGBD 数据实时采集（部分 TOF 模块支持此功能）
9. 支持 IMU 数据实时采集（部分 TOF 模块支持此功能）
10. 支持 Windows7, Linux(Ubuntu…), Android…

本 SDK 支持的 TOF 模块产品有：

1、EPC
2、MARS01A
3、MARS01B
4、MARS04A
5、MARS04B
6、MARS05
7、MARS05A
8、其他未一一罗列产品

1.2 本 SDK 包含的内容

- 指定平台的头文件、库文件、配置文件；
- API 使用说明文档【本文档】；
- SDK 的样例代码；
- 采用 CMake 的编译脚本；

1.3 支持平台

- Windows7、10 x64
- Ubuntu16.04 x64
- ARM
- Android8.0，9.0

2 TOF 模块 SDK 接口说明

2.1 TOFD_Init

函数原型：

TOFDDLL TOFRET TOFD_Init(TofDevInitParam* pInitParam);

函数描述：

初始化 TOF 设备，TOF SDK 的其他函数的调用必须在此函数调用之后才能调用。

函数参数：

pInitParam	【输入】	SDK 库初始化参数， 不能为 NULL。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.2 TOFD_Uninit

函数原型：

TOFDDLL TOFRET TOFD_Uninit(void);

函数描述：

停止所有数据的输出（包含 TOF 的点云数据和 IR 图像数据，RGB 数据、IMU 数据）并且清除相关的内存资源。此函数通常是在应用的主函数退出时调用。在此函数调用之后，TOF 模块 SDK 的其他函数的不能被调用。

2.3 TOFD_GetSDKVersion

函数原型：

TOFDDLL SCHAR* TOFD_GetSDKVersion(void);

函数描述：

获取 SDK 的版本信息（返回值为字符串型版本号）。

返回值：

函数执行成功返回 SDK 的版本信息.

2.4 TOFD_SearchDevice

函数原型：

TOFDDLL TOFRET TOFD_SearchDevice(TofDeviceDescriptor **ppDevsDesc, UINT32* pDevNum);

函数描述：

搜索连接到本系统上的所有 TOF 设备。

函数参数：

ppDevsDesc	【输出】	请参考第 4 章中的结构体 TofDeviceDescriptor 的内容， 不能为 NULL。
pDevNum	【输出】	设备个数，不能为 NULL。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.5 TOFD_OpenDevice

函数原型：

TOFDDLL HTOFD TOFD_OpenDevice(TofDeviceDescriptor *pDevDesc, FNTofDeviceStatus fnTofDevStatus, void* pUserData);

函数描述：

此函数用来打开一个 TOF 模块设备，对于设备的操作函数，包括 TOF 的点云和 IR 图像获取、RGB 图像获取， RGBD 图像获取， IMU 数据获取，TOF 的曝光、以及过滤器的设置等，都必须在此函数被调用之后执行。

函数参数：

pDevDesc	【输入】	请参考第 4 章中的结构体 TofDeviceDescriptor 的内容，此参数只能是通过 TOFD_SearchDevice 函数获取到后传入，不能为 NULL。
fnTofDevStatus	【输入】	请参考第 4 章中的回调函数 FNTofDeviceStatus 的内容，TOF 模块设备的状态回调函数，当设备状态改变时则通过此函数通知到应用程序，此参数可以为 NULL。
pUserData	【输入】	用户数据指针，此参数将作为 fnTofDevStatus 的参数传递给上层应用。

返回值：

函数执行成功返回设备的句柄，否则返回 NULL.

2.6 TOFD_OpenDevice_WithFd

函数原型：

TOFDDLL HTOFD TOFD_OpenDevice_WithFd(TofDeviceDescriptorWithFd *pDevDesc, FNTofDeviceStatus fnTofDevStatus, void* pUserData);

函数描述：

此函数用来打开一个 TOF 模块设备，对于设备的操作函数，包括 TOF 的点云和 IR 图像获取、RGB 图像获取， RGBD 图像获取， IMU 数据获取，TOF 的曝光、以及过滤器的设置等，都必须在此函数被调用之后执行。

函数参数：

pDevDesc	【输入】	请参考第 4 章中的结构体 TofDeviceDescriptor 的内容，此参数只能是通过 TOFD_SearchDevice 函数获取到后传入，不能为 NULL。
fnTofDevStatus	【输入】	请参考第 4 章中的回调函数 FNTofDeviceStatus 的内容，TOF 模块设备的状态回调函数，当设备状态改变时则通过此函数通知到应用程序，此参数可以为 NULL。
pUserData	【输入】	用户数据指针，此参数将作为 fnTofDevStatus 的参数传递给上层应用。

返回值：

函数执行成功返回设备的句柄，否则返回 NULL.

2.7 TOFD_CloseDevice

函数原型：

TOFDDLL TOFRET TOFD_CloseDevice(HTOFD hTofDev);

函数描述：

关闭 TOF 设备，对于设备的操作函数，包括 TOF 的点云和 IR 图像获取、RGB 图像获取，RGBD 图像获取， IMU 数据获取，TOF 的曝光、以及过滤器的设置等，都必须在此函数被调用之前执行。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.8 TOFD_GetDeviceInfo

函数原型：

TOFDDLL TOFRET TOFD_GetDeviceInfo(HTOFD hTofDev, TofDeviceInfo *pTofDeviceInfo);

函数描述：

获取设备信息（一般表示设备能力）。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
pTofDeviceInfo	【输出】	获取到的获取设备信息，只读。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.9 TOFD_GetDeviceParam

函数原型：

TOFDDLL TOFRET TOFD_GetDeviceParam(HTOFD hTofDev, TofDeviceParam *pTofDeviceParam);

函数描述：

获取设备参数（需要设备支持）（已逐步废弃）。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
pTofDeviceParam	【输出】	获取到的设备参数。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.10 TOFD_SetDeviceParam

函数原型：

TOFDDLL TOFRET TOFD_SetDeviceParam(HTOFD hTofDev, TofDeviceParam *pTofDeviceParam);

函数描述：

设置设备参数（需要设备支持）（已逐步废弃）。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
pTofDeviceParam	【输入】	需要设置到设备的参数。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.11 TOFD_GetDeviceParamV20

函数原型：

TOFDDLL TOFRET TOFD_GetDeviceParamV20(HTOFD hTofDev, TofDeviceParamV20 *pTofDeviceParam);

函数描述：

获取指定类型的设备参数（2.0 版本接口）。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。

pTofDeviceParam	【输入/输出】	获取到的获取设备参数。 其中的 type 为输入参数，uParam 为输出参数； 1） 当 type 为 TOF_DEV_PARAM_Temperature 时，uParam 中的 struTemperature 有效； 2） 当 type 为 TOF_DEV_PARAM_TofLensParameter 时，uParam 中的 struTofLensParameter 有效； 3） 当 type 为 TOF_DEV_PARAM_TofCalibData 时，uParam 中的 struTofCalibData 有效； 4） 当 type 为 TOF_DEV_PARAM_netdevinfo 时，uParam 中的 stuNetDevData 有效

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.12 TOFD_SetDeviceParamV20

函数原型：

TOFDDLL TOFRET TOFD_SetDeviceParamV20(HTOFD hTofDev, TofDeviceParamV20 *pTofDeviceParam);

函数描述：

设置指定类型的设备参数（2.0 版本接口）。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
pTofDeviceParam	【输入/输出】	获取到的获取设备参数。 其中的 type 为输入参数，uParam 为输出参数； 5） 当 type 为 TOF_DEV_PARAM_Temperature 时，uParam 中的 struTemperature 有效； 6） 当 type 为 TOF_DEV_PARAM_TofLensParameter 时，uParam 中的 struTofLensParameter 有效； 7） 当 type 为 TOF_DEV_PARAM_TofCalibData 时，uParam 中的 struTofCalibData 有效； 8） 当 type 为 TOF_DEV_PARAM_netdevinfo 时，uParam 中的 stuNetDevData 有效

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.13 TOFD_SetTofAE

函数原型：

TOFDDLL TOFRET TOFD_SetTofAE(HTOFD hTofDev, const SBOOL bEnable);

函数描述：

设置 TOF 曝光模式，手动模式或者自动模式。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
bEnable	【输入】	是否是自动曝光模式，true 为自动曝光模式，false 为手动曝光模式。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.14 TOFD_SetTofExpTime

函数原型：

TOFDDLL TOFRET TOFD_SetTofExpTime(HTOFD hTofDev, const UINT32 expTime);

函数描述：

设置 TOF 当前的曝光时间。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
expTime	【输入】	TOF 曝光时间。此参数必须在 TOF 有效曝光时间范围内，有效曝光时间通过 TOFD_GetTofExpTime 函数获取。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.15 TOFD_GetTofExpTime

函数原型：

TOFDDLL TOFRET TOFD_GetTofExpTime(HTOFD hTofDev, TofExpouse *pExp);

函数描述：

获取 TOF 的曝光时间参数。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
pExp	【输出】	TOF 曝光时间参数，不能为 NULL。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.16 TOFD_SetTofFilter

函数原型：

TOFDDLL TOFRET TOFD_SetTofFilter(HTOFD hTofDev, const TOF_FILTER type, const SBOOL bEnable);

函数描述：

打开或关闭指定类型的 TOF 过滤器。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
type	【输入】	滤波类型。
bEnable	【输入】	是否打开 TOF 滤波，true 为打开，false 为关闭。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.17 TOFD_GetTofFilter

函数原型：

TOFDDLL TOFRET TOFD_GetTofFilter(HTOFD hTofDev, const TOF_FILTER type, SBOOL* pbEnable);

函数描述：

获取指定类型的 TOF 过滤器的开关状态。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
type	【输入】	滤波类型。
pbEnable	【输出】	是否打开了 TOF 滤波，true 为打开，false 为关闭。，不能为 NULL。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.18 TOFD_SetTofHDRZ

函数原型：

TOFDDLL TOFRET TOFD_SetTofHDRZ(HTOFD hTofDev, const SBOOL bEnable);

函数描述：

打开或关闭 TOF HDRZ。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
bEnable	【输入】	是否打开 TOF HDRZ，true 为打开，false 为关闭。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.19 TOFD_SetTofRemoveINS

函数原型：

TOFDDLL TOFRET TOFD_SetTofRemoveINS(HTOFD hTofDev, const SBOOL bEnable);

函数描述：

打开或关闭 TOF RemoveINS 算法。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
bEnable	【输入】	是否打开 TOF RemoveINS，true 为打开，false 为关闭。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.20 TOFD_SetTofMPIFlag

函数原型：

TOFDDLL TOFRET TOFD_SetTofMPIFlag(HTOFD hTofDev, const SBOOL bEnable);

函数描述：

打开或关闭 TOF MPIFlag 算法（已废弃，请使用 TOFD_SetTofFilter(xxx, TOF_FILTER_MPIFilter, xxx)）。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
bEnable	【输入】	是否打开 TOF MPIFlag，true 为打开，false 为关闭。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.21 TOFD_StartTofStream

函数原型：

TOFDDLL TOFRET TOFD_StartTofStream(HTOFD hTofDev, const TOF_MODE tofMode, FNTofStream fnTofStream, void* pUserData);

函数描述：

启动实时获取 TOF 点云数据、 IR 图像数据。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
tofMode	【输入】	TOF 模式， 此参数通过 TOFD_SearchDevice 函数获取，请参考第 4 章中枚举 TOF_MODE 部分描述。
fnTofStream	【输入】	输出 TOF 点云数据、 IR 图像数据的回调函数。此参数不能为 NULL。
pUserData	【输入】	用户数据指针，此参数将当作 fnTofStream 的一个参数被输出到应用程序。

返回值：

函数执行成功返回 TOFRET_SUCCESS 或者 TOFRET_SUCCESS_READING_CALIB，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

特别注意：当返回值为 TOFRET_SUCCESS_READING_CALIB 时，直到数据流回调出的过程中，一般的会通过 FNTofDeviceStatus 回调出状态；

2.22 TOFD_StopTofStream

函数原型：

TOFDDLL TOFRET TOFD_StopTofStream(HTOFD hTofDev);

函数描述：

停止实时获取 TOF 点云数据、 IR 图像数据。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.23 TOFD_GetRgbProperty

函数原型：

TOFDDLL TOFRET TOFD_GetRgbProperty(HTOFD hTofDev, const RgbVideoControlProperty Property, RgbVideoControl *pValue);

函数描述：

获取 RGB 模组的指定属性的参数值。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
Property	【输入】	指定的 RGB 属性。
pValue	【输出】	获取到的指定的 RGB 属性的参数值。此参数不能为 NULL。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.24 TOFD_SetRgbProperty

函数原型：

TOFDDLL TOFRET TOFD_SetRgbProperty(HTOFD hTofDev, const RgbVideoControlProperty Property, const SINT32 lValue, const RgbVideoControlFlags lFlag);

函数描述：

设置 RGB 模组的指定属性的参数值。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
Property	【输入】	指定的 RGB 属性。
lValue	【输入】	设置的指定的 RGB 属性的参数值。
lFlag	【输入】	设置的指定的 RGB 属性的参数值的附件属性，用于表明自动还是手动。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.25 TOFD_StartRgbStream

函数原型：

TOFDDLL TOFRET TOFD_StartRgbStream(HTOFD hTofDev, FNRgbStream fnRgbStream, void* pUserData);

函数描述：

启动实时获取 RGB 图像数据。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
fnRgbStream	【输入】	输出 RGB 数据的回调函数。此参数不能为 NULL。

pUserData	【输入】	用户数据指针，此参数将当作 fnRgbStream 的一个参数被输出到应用程序。

返回值：

函数执行成功返回 TOFRET_SUCCESS 或者 TOFRET_SUCCESS_READING_CALIB，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

特别注意：当返回值为 TOFRET_SUCCESS_READING_CALIB 时，直到数据流回调出的过程中，一般的会通过 FNTofDeviceStatus 回调出状态；

2.26 TOFD_StopRgbStream

函数原型：

TOFDDLL TOFRET TOFD_StopRgbStream(HTOFD hTofDev);

函数描述：

停止实时获取 RGB 图像数据。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.27 TOFD_StartImuStream

函数原型：

TOFDDLL TOFRET TOFD_StartImuStream(HTOFD hTofDev, FNImuStream fnImuStream, void* pUserData);

函数描述：

启动实时获取 IMU 数据。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。
fnImuStream	【输入】	输出 IMU 数据的回调函数。此参数不能为 NULL。
pUserData	【输入】	用户数据指针，此参数将当作 fnImuStream 的一个参数被输出到应用程序。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

2.28 TOFD_StopImuStream

函数原型：

TOFDDLL TOFRET TOFD_StopImuStream(HTOFD hTofDev);

函数描述：

停止实时获取 IMU 数据。

函数参数：

hTofDev	【输入】	TOF 设备句柄，不能为 NULL。

返回值：

函数执行成功返回 TOFRET_SUCCESS，否则返回其他错误值,具体错误值请参考 TOFRET 部分描述.

3 公用数据结构与类型定义

3.1 TOFRET

原型：

```
typedef enum tagTOFRET
{
        /** Success (no error) */
        TOFRET_SUCCESS = 0x00000000,
        /** Success (no error, and start to read calib data) */
        TOFRET_SUCCESS_READING_CALIB = 0x00000001,



         /** Input/output error */
        TOFRET_ERROR_IO = 0x80000001,
         /** Invalid parameter */
        TOFRET_ERROR_INVALID_PARAM = 0x80000002,
        /** Access denied (insufficient permissions) */
        TOFRET_ERROR_ACCESS = 0x80000003,
        /** No such device (it may have been disconnected) */
        TOFRET_ERROR_NO_DEVICE = 0x80000004,
        /** Operation timed out */
        TOFRET_ERROR_TIMEOUT = 0x80000005,
        /** Overflow */
        TOFRET_ERROR_OVERFLOW = 0x80000006,
        /** Insufficient memory */
        TOFRET_ERROR_NO_MEM = 0x80000007,
        /** Operation not supported or unimplemented on this platform */
        TOFRET_ERROR_WRONG_STATUS    = 0x80000008,
        /** Operation not supported */
        TOFRET_ERROR_NOT_SUPPORTED = 0x80000009,
        /** Device is in use now */
        TOFRET_ERROR_ALREADY_IN_USE = 0x8000000A,
        /** Error Data */
        TOFRET_ERROR_DATA = 0x8000000B,
        /** Cfg file not found */
        TOFRET_ERROR_CFG_FILE_NOT_FOUND = 0x8000000C,
        /** Read Calib falied */
        TOFRET_ERROR_READ_CALIB_FAILED = 0x8000000D,

        /** USB write error */
        TOFRET_ERROR_USB_WRITE = 0x80010001,
        /** USB read error */
        TOFRET_ERROR_USB_READ = 0x80010002,
        /** USB disconnect */
        TOFRET_ERROR_USB_DISCONNECT = 0x80010003,

        /* generic fail */
        TOFRET_HAL_FAILED = 0x80060001,
        /* operation not support */
        TOFRET_HAL_UNSUPPORT = 0x80060002,
        /* device is unreponsive */
        TOFRET_HAL_HARDWARE_UNRESPONSIVE = 0x80060003,
        /* timeout */
```

```
    TOFRET_HAL_TIMEOUT = 0x80060004,
    /* interface board not support */
    TOFRET_HAL_INTERFACE_BOARD_NOT_SUPPORT = 0x80060005,
    /* configuration read error */
    TOFRET_HAL_CONFIG_READ_FAILED = 0x80060006,
    /* module dll load failed */
    TOFRET_HAL_MODULE_LOAD_FAILED = 0x80060007,
    /* call module dll function failed */
    TOFRET_HAL_MODULE_SYSMBOL_CALL_FAILED = 0x80060008,
    /* object instance failed */
    TOFRET_HAL_OBJ_INSTANCE_FAILED = 0x80060009,
    /* not found camera */
    TOFRET_HAL_CAMERA_NOT_FOUND = 0x8006000A,
    /* platform setting failed */
    TOFRET_HAL_INTERFACE_BOARD_SETTING_FAILED = 0x8006000B,
    /* iic read failed */
    TOFRET_HAL_IIC_READ_FAILED = 0x8006000C,
    /* iic write failed */
    TOFRET_HAL_IIC_WRITE_FAILED = 0x8006000D,
    /* io operation failed */
    TOFRET_HAL_IO_SETTING_FAILED = 0x8006000E,

    /** Other error */
    TOFRET_ERROR_OTHER = 0x88100001,
}TOFRET;
```

描述：

TOF SDK 错误值定义。

参数：

TOFRET_SUCCESS	成功
TOFRET_SUCCESS_READING_CALIB	成功，并且开始了读取标定文件
TOFRET_ERROR_IO	IO 错误
TOFRET_ERROR_INVALID_PARAM	无效的参数
TOFRET_ERROR_ACCESS	设备操作失败

TOFRET_ERROR_NO_DEVICE	设备不存在
TOFRET_ERROR_TIMEOUT	访问设备超时
TOFRET_ERROR_OVERFLOW	数据出现溢出
TOFRET_ERROR_NO_MEM	分配内存失败
TOFRET_ERROR_WRONG_STATUS	状态错误
TOFRET_ERROR_NOT_SUPPORTED	不支持的功能；
TOFRET_ERROR_ALREADY_IN_USE	设备应经在使用中，表示设备已经被打开

TOFRET_ERROR_DATA	错误的数据
TOFRET_ERROR_CFG_FILE_NOT_FOUND	未找到配置文件
TOFRET_ERROR_READ_CALIB_FAILED	标定文件读取失败
TOFRET_ERROR_USB_WRITE	USB 写错误
TOFRET_ERROR_USB_READ	USB 读错误
TOFRET_ERROR_USB_DISCONNECT	USB 断开连接
TOFRET_HAL_FAILED	访问 HAL 层失败
TOFRET_HAL_UNSUPPORT	HAL 层不支持此功能
TOFRET_HAL_HARDWARE_UNRESPONSIVE	硬件对 HAL 层没有反应
TOFRET_HAL_TIMEOUT	HAL 访问硬件超时
TOFRET_HAL_INTERFACE_BOARD_NOT_SUPPORT	HAL 接口板不支持
TOFRET_HAL_CONFIG_READ_FAILED	HAL 层读配置错误
TOFRET_HAL_MODULE_LOAD_FAILED	模组打开失败
TOFRET_HAL_MODULE_SYSMBOL_CALL_FAILED	HAL 层符号表调用失败
TOFRET_HAL_OBJ_INSTANCE_FAILE	HAL 层目标初始化错误
TOFRET_HAL_CAMERA_NOT_FOUND	HAL 层 TOF 设备没有发现
TOFRET_HAL_INTERFACE_BOARD_SETTING_FAILED	HAL 层接口板设置失败
TOFRET_HAL_IIC_READ_FAILED	HAL 层 I2C 读失败
TOFRET_HAL_IIC_WRITE_FAILED	HAL 层 I2C 写失败
TOFRET_HAL_IO_SETTING_FAILED	HAL 层设置失败
TOFRET_ERROR_OTHER	其他错误

3.2 MAKE_UNIQUE_ID

原型：

```
#define MAKE_UNIQUE_ID(major, sub, a, b) ((major<<24) | (sub<<16) | (a<<8) | (b))
```

描述：

生成特定规则的 32 位 ID 号.

3.3 TOF_MODE

原型：

```
typedef enum tagTOF_MODE
{
        //双频
```

```
    TOF_MODE_STERO_5FPS = 0x00000001,
    TOF_MODE_STERO_10FPS = 0x00000002,
    TOF_MODE_STERO_15FPS = 0x00000004,
    TOF_MODE_STERO_30FPS = 0x00000008,
    TOF_MODE_STERO_45FPS = 0x00000010,
    TOF_MODE_STERO_60FPS = 0x00000020,

    //单频
    TOF_MODE_MONO_5FPS = 0x00000040,
    TOF_MODE_MONO_10FPS = 0x00000080,
    TOF_MODE_MONO_15FPS = 0x00000100,
    TOF_MODE_MONO_30FPS = 0x00000200,
    TOF_MODE_MONO_45FPS = 0x00000400,
    TOF_MODE_MONO_60FPS = 0x00000800,

    //HDRZ：这几个模式代表具有raw数据的HDRZ融合的
    TOF_MODE_HDRZ_5FPS = 0x00001000,
    TOF_MODE_HDRZ_10FPS = 0x00002000,
    TOF_MODE_HDRZ_15FPS = 0x00004000,
    TOF_MODE_HDRZ_30FPS = 0x00008000,
    TOF_MODE_HDRZ_45FPS = 0x00010000,
    TOF_MODE_HDRZ_60FPS = 0x00020000,

    //帧率不同
    TOF_MODE_5FPS      = 0x00040000,
    TOF_MODE_10FPS     = 0x00080000,
    TOF_MODE_20FPS     = 0x00100000,
    TOF_MODE_30FPS     = 0x00200000,
    TOF_MODE_45FPS     = 0x00400000,
    TOF_MODE_60FPS     = 0x00800000,

    //ADI特定
    TOF_MODE_ADI_1M5   = 0x01000000,
    TOF_MODE_ADI_5M    = 0x02000000,

    //自定义
    TOF_MODE_CUSTOM_1  = 0x04000000,
    TOF_MODE_CUSTOM_2  = 0x08000000,
    TOF_MODE_CUSTOM_3  = 0x10000000,
    TOF_MODE_CUSTOM_4  = 0x20000000,
    TOF_MODE_CUSTOM_5  = 0x40000000,

    //DEBUG模式
    TOF_MODE_DEBUG     = 0x80000000,

}TOF_MODE;
```

描述：

TOF_MODE 枚举定义 TOF 模式.

类型：

TOF_MODE_STERO_5FPS	Double frequency, 5 fps

TOF_MODE_STERO_10FPS	Double frequency, 10 fps
TOF_MODE_STERO_15FPS	Double frequency, 15 fps
TOF_MODE_STERO_30FPS	Double frequency, 30 fps
TOF_MODE_STERO_45FPS	Double frequency, 45 fps
TOF_MODE_STERO_60FPS	Double frequency, 60 fps
TOF_MONO_STERO_5FPS	Single frequency, 5 fps
TOF_MONO_STERO_10FPS	Single frequency, 10 fps
TOF_MONO_STERO_15FPS	Single frequency, 15 fps
TOF_MONO_STERO_30FPS	Single frequency, 30 fps
TOF_MONO_STERO_45FPS	Single frequency, 45 fps
TOF_MONO_STERO_60FPS	Single frequency, 60 fps
TOF_MODE_HDRZ_5FPS	HDRZ frequency, 5 fps
TOF_MODE_HDRZ_10FPS	HDRZ frequency, 10 fps
TOF_MODE_HDRZ_15FPS	HDRZ frequency, 15 fps
TOF_MODE_HDRZ_30FPS	HDRZ frequency, 30 fps
TOF_MODE_HDRZ_45FPS	HDRZ frequency, 45 fps
TOF_MODE_HDRZ_60FPS	HDRZ frequency, 60 fps
TOF_MODE_5FPS	5 fps
TOF_MODE_10FPS	10 fps
TOF_MODE_20FPS	20fps
TOF_MODE_30FPS	30 fps
TOF_MODE_45FPS	45 fps
TOF_MODE_60FPS	60 fps
TOF_MODE_ADI_5M	ADI 5M mode
TOF_MODE_ADI_1M5	ADI 1M5 mode
TOF_MODE_CUSTOM_1	custom mode 1
TOF_MODE_CUSTOM_2	custom mode 2
TOF_MODE_CUSTOM_3	custom mode 3
TOF_MODE_CUSTOM_4	custom mode 4
TOF_MODE_CUSTOM_5	custom mode 5
TOF_MODE_DEBUG	debug mode

3.4 TOF_FILTER

原型：

```
typedef enum tagTOF_FILTER
{
        TOF_FILTER_RemoveFlyingPixel = 0x00000001,
```

```
    TOF_FILTER_AdaptiveNoiseFilter = 0x00000002,
    TOF_FILTER_InterFrameFilter = 0x00000004,
    TOF_FILTER_PointCloudFilter = 0x00000008,
    TOF_FILTER_StraylightFilter = 0x00000010,
    TOF_FILTER_CalcIntensities = 0x00000020,
    TOF_FILTER_MPIFlagAverage = 0x00000040,
    TOF_FILTER_MPIFlagAmplitude = 0x00000080,
    TOF_FILTER_MPIFlagDistance = 0x00000100,
    TOF_FILTER_ValidateImage = 0x00000200,
    TOF_FILTER_SparsePointCloud = 0x00000400,
    TOF_FILTER_Average = 0x00000800,
    TOF_FILTER_Median = 0x00001000,
    TOF_FILTER_Confidence = 0x00002000,
    TOF_FILTER_MPIFilter = 0x00004000,
    TOF_FILTER_PointCloudCorrect = 0x00008000,
    TOF_FILTER_LineRecognition = 0x00010000,
    TOF_FILTER_RadialFusion      = 0x00020000,
}TOF_FILTER;
```

描述：

TOF 数据滤波种类。

类型：

TOF_FILTER_RemoveFlyingPixel	移除飞点滤波
TOF_FILTER_AdaptiveNoiseFilter	自适应噪声滤波
TOF_FILTER_InterFrameFilter	帧间滤波
TOF_FILTER_PointCloudFilter	点云滤波
TOF_FILTER_StraylightFilter	杂光滤波
TOF_FILTER_CalcIntensities	CalcIntensities 滤波
TOF_FILTER_MPIFlagAverage	MPIFlagAverage 滤波
TOF_FILTER_MPIFlagAmplitude	MPIFlagAmplitude 滤波
TOF_FILTER_MPIFlagDistance	MPIFlagDistance 滤波
TOF_FILTER_ValidateImage	ValidateImage 滤波
TOF_FILTER_SparsePointCloud	稀疏点云滤波
TOF_FILTER_Average	均值滤波
TOF_FILTER_Median	中值滤波
TOF_FILTER_Confidence	Confidence 滤波
TOF_FILTER_MPIFilter	MPI 滤波
TOF_FILTER_PointCloudCorrect	点云矫正滤波
TOF_FILTER_LineRecognition	黑线检测滤波
TOF_FILTER_RadialFusion	射线融合滤波

3.5 TofFilterCfg_RemoveFlyingPixel

原型：

```
typedef struct tagTofFilterCfg_RemoveFlyingPixel
{
        FLOAT32 f0;
        FLOAT32 f1;
        FLOAT32 nd;
        FLOAT32 fd;
}TofFilterCfg_RemoveFlyingPixel;
```

描述：

移除飞点滤波参数。

类型：

f0	
f1	
nd	
fd	

3.6 TofFilterCfg_AdaptiveNoiseFilter

原型：

```
typedef struct tagTofFilterCfg_AdaptiveNoiseFilter
{
        SINT32  k;
        FLOAT32 s;
        SINT32  t;
}TofFilterCfg_AdaptiveNoiseFilter;
```

描述：

自适应噪声滤波参数。

类型：

k	
s	
t	

3.7 TofFilterCfg_InterFrameFilter

原型：

```
typedef struct tagTofFilterCfg_InterFrameFilter
{
        FLOAT32 mdg;
        FLOAT32 mdt;
        FLOAT32 fg1;
        FLOAT32 fg2;
}TofFilterCfg_InterFrameFilter;
```

描述：

帧间滤波参数。

类型：

mdg	
mdt	
fg1	
fg2	

3.8 TofFilterCfg_PointCloudFilter

原型：

```
typedef struct tagTofFilterCfg_PointCloudFilter
{
        SINT32 k;
}TofFilterCfg_PointCloudFilter;
```

描述：

点云滤波参数。

类型：

k	

3.9 TofFilterCfg_StraylightFilter

原型：

```
typedef struct tagTofFilterCfg_StraylightFilter
{
        FLOAT32 d[16];
        FLOAT32 t[16];
}TofFilterCfg_StraylightFilter;
```

描述：

杂光滤波参数。

类型：

d	
t	

3.10 TofFilterCfg_CalcIntensities

原型：

```
typedef struct tagTofFilterCfg_CalcIntensities
{
        UINT8 szRes[4];//预留,4 字节对齐
}TofFilterCfg_CalcIntensities;
```

描述：

CalcIntensities 滤波参数。

类型：

szRes	预留

3.11 TofFilterCfg_MPIFlagAverage

原型：

```
typedef struct tagTofFilterCfg_MPIFlagAverage
{
        UINT8 szRes[4];//预留,4 字节对齐
}TofFilterCfg_MPIFlagAverage;
```

描述：

MPIFlagAverage 滤波参数。

类型：

szRes	预留

3.12 TofFilterCfg_MPIFlagAmplitude

原型：

```
typedef struct tagTofFilterCfg_MPIFlagAmplitude
{
        FLOAT32 mat;
        FLOAT32 ndt;
}TofFilterCfg_MPIFlagAmplitude;
```

描述：

MPIFlagAmplitude 滤波参数。

类型：

mat	
ndt	

3.13 TofFilterCfg_MPIFlagDistance

原型：

```
typedef struct tagTofFilterCfg_MPIFlagDistance
{
        UINT8 szRes[4];//预留,4 字节对齐
}TofFilterCfg_MPIFlagDistance;
```

描述：

MPIFlagDistance 滤波参数。

类型：

szRes	预留

3.14 TofFilterCfg_ValidateImage

原型：

```
typedef struct tagTofFilterCfg_ValidateImage
{
        UINT8 szRes[4];//预留,4 字节对齐
}TofFilterCfg_ValidateImage;
```

描述：

ValidateImage 滤波参数。

类型：

szRes	预留

3.15 TofFilterCfg_SparsePointCloud

原型：

```
typedef struct tagTofFilterCfg_SparsePointCloud
{
        UINT8 szRes[4];//预留,4 字节对齐
}TofFilterCfg_SparsePointCloud;
```

描述：

稀疏点云滤波参数。

类型：

szRes	预留

3.16 TofFilterCfg_Average

原型：

```
typedef struct tagTofFilterCfg_Average
{
        UINT8 szRes[4];//预留,4 字节对齐
}TofFilterCfg_Average;
```

描述：

均值滤波参数。

类型：

szRes	预留

3.17 TofFilterCfg_Median

原型：

```
typedef struct tagTofFilterCfg_Median
{
        UINT8 szRes[4];//预留,4 字节对齐
}TofFilterCfg_Median;
```

描述：

中值滤波参数。

类型：

szRes	预留

3.18 TofFilterCfg_Confidence

原型：

```
typedef struct tagTofFilterCfg_Confidence
{
        FLOAT32 t;
}TofFilterCfg_Confidence;
```

描述：

Confidence 滤波参数。

类型：

t	

3.19 TofFilterCfg_MPIFilter

原型：

```
typedef struct tagTofFilterCfg_MPIFilter
{
        FLOAT32 ndt;
        FLOAT32 fdt;
        FLOAT32 nnr;
        FLOAT32 mnr;
        FLOAT32 fnr;
        FLOAT32 rd;
}TofFilterCfg_MPIFilter;
```

描述：

MPI 滤波参数。

类型：

ndt	
fdt	
nnr	
mnr	
fnr	
rd	

3.20 TofFilterCfg_PointCloudCorrect

原型：

```
typedef struct tagTofFilterCfg_PointCloudCorrect
{
        FLOAT32 da;//模组下斜的角度
        FLOAT32 tgd;//模组距地面的高度
        FLOAT32 t1;
        FLOAT32 t2;
}TofFilterCfg_PointCloudCorrect;
```

描述：

点云矫正滤波参数。

类型：

da	
tgd	
t1	
T2	

3.21 TofFilterCfg_ LineRecognition

原型：

```
typedef struct tagTofFilterCfg_LineRecognition
{
        SINT32 ht;
        SINT32 cgt;
        SINT32 fgst;
        FLOAT32 gstr;
        SINT32 spgt;
        SINT32 opgt;
        SINT32 rc;
        SINT32 lc;
        SINT32 ma;
}TofFilterCfg_LineRecognition;
```

描述：

黑线识别参数。

类型：

ht	
cgt	
fgst	

gstr	
spgt	
opgt	
rc	
lc	
ma	

3.22 TofFilterCfg_RadialFusion

原型：

```
typedef struct tagTofFilterCfg_RadialFusion
{
        UINT8 szRes[4];//预留,4 字节对齐
}TofFilterCfg_RadialFusion;
```

描述：

黑线识别参数。

类型：

szRes	预留,4 字节对齐

3.23 TofFilterCfg

原型：

```
typedef struct tagTofFilterCfg
{
        TOF_FILTER type;//某一种滤波类型，一般是输入参数（只读）

        union
        {
                SBOOL bEnable;//是否启用，可以是输入或者输出参数(暂时不开放,目前属于无效字段)
                UINT8 szRes[4];//预留,4 字节对齐
        }uRes;

        union
        {
                TofFilterCfg_RemoveFlyingPixel struRemoveFlyingPixel;//当type取值为TOF_FILTER_RemoveFlyingPixel时有效
                TofFilterCfg_AdaptiveNoiseFilter struAdaptiveNoiseFilter;//当type取值为
```

TOF_FILTER_AdaptiveNoiseFilter时有效
TofFilterCfg_InterFrameFilter struInterFrameFilter;//当type取值为TOF_FILTER_InterFrameFilter时有效
TofFilterCfg_PointCloudFilter struPointCloudFilter;//当type取值为TOF_FILTER_PointCloudFilter时有效
TofFilterCfg_StraylightFilter struStraylightFilter;//当type取值为TOF_FILTER_StraylightFilter时有效
TofFilterCfg_CalcIntensities struCalcIntensities;//当type取值为TOF_FILTER_CalcIntensities时有效
TofFilterCfg_MPIFlagAverage struMPIFlagAverage;//当type取值为TOF_FILTER_MPIFlagAverage时有效
TofFilterCfg_MPIFlagAmplitude struMPIFlagAmplitude;//当type取值为TOF_FILTER_MPIFlagAmplitude时有效
TofFilterCfg_MPIFlagDistance struMPIFlagDistance;//当type取值为TOF_FILTER_MPIFlagDistance时有效
TofFilterCfg_ValidateImage struValidateImage;//当type取值为TOF_FILTER_ValidateImage时有效
TofFilterCfg_SparsePointCloud struSparsePointCloud;//当type取值为TOF_FILTER_SparsePointCloud时有效
TofFilterCfg_Average struAverage;//当type取值为TOF_FILTER_Average时有效
TofFilterCfg_Median struMedian;//当type取值为TOF_FILTER_Median时有效
TofFilterCfg_Confidence struConfidence;//当type取值为TOF_FILTER_Confidence时有效
TofFilterCfg_MPIFilter struMPIFilter;//当type取值为TOF_FILTER_MPIFilter时有效
TofFilterCfg_PointCloudCorrect struPointCloudCorrect;//当type取值为TOF_FILTER_PointCloudCorrect时有效
TofFilterCfg_LineRecognition struLineRecognition;//当type取值为TOF_FILTER_LineRecognition时有效
TofFilterCfg_RadialFusion struRadialFusion;//当type取值为TOF_FILTER_RadialFusion时有效
}uCfg;//某种滤波的具体配置，可以是输入或者输出参数

}TofFilterCfg;

描述：

详细的滤波参数配置。

类型：

type		某一种滤波类型，一般是输入参数（只读）
uRes	bEnable	是否启用，可以是输入或者输出参数(暂时不开放,目前属于无效字段)
	szRes	预留,4 字节对齐
uCfg		由 type 指定的滤波类型具体参数配置，根据 type 的不同，使用不同的字段；
	struRemoveFlyingPixel	当 type 取值为 TOF_FILTER_RemoveFlyingPixel 时有效

	struAdaptiveNoiseFilter	当 type 取值为 TOF_FILTER_AdaptiveNoiseFilter 时有效
	struInterFrameFilter	当 type 取值为 TOF_FILTER_InterFrameFilter 时有效
	struPointCloudFilter	当 type 取值为 TOF_FILTER_PointCloudFilter 时有效
	struStraylightFilter	当 type 取值为 TOF_FILTER_StraylightFilter 时有效
	struCalcIntensities	当 type 取值为 TOF_FILTER_CalcIntensities 时有效
	struMPIFlagAverage	当 type 取值为 TOF_FILTER_MPIFlagAverage 时有效
	struMPIFlagAmplitude	当 type 取值为 TOF_FILTER_MPIFlagAmplitude 时有效
	struMPIFlagDistance	当 type 取值为 TOF_FILTER_MPIFlagDistance 时有效
	struValidateImage	当 type 取值为 TOF_FILTER_ValidateImage 时有效
	struSparsePointCloud	当 type 取值为 TOF_FILTER_SparsePointCloud 时有效
	struAverage	当 type 取值为 TOF_FILTER_Average 时有效
	struMedian	当 type 取值为 TOF_FILTER_Median 时有效
	struConfidence	当 type 取值为 TOF_FILTER_Confidence 时有效
	struMPIFilter	当 type 取值为 TOF_FILTER_MPIFilter 时有效
	struPointCloudCorrect	当 type 取值为 TOF_FILTER_PointCloudCorrect 时有效
	struLineRecognition	当 type 取值为 TOF_FILTER_LineRecognition 时有效
	struRadialFusion	当 type 取值为 TOF_FILTER_RadialFusion 时有效

3.24 EXP_MODE

原型：

```
typedef enum tagEXP_MODE
{
        EXP_MODE_MANUAL = 0x00000001,//手动曝光
        EXP_MODE_AUTO = 0x00000002,//自动曝光(AE)
}EXP_MODE;
```

描述：

TOF 曝光类型。

类型：

EXP_MODE_MANUAL	手动曝光
EXP_MODE_AUTO	自动曝光(AE)

3.25 GRAY_FORMAT

原型：

```
typedef enum tagGRAY_FORMAT
{
        GRAY_FORMAT_UINT8 = 0,//8 位数据
        GRAY_FORMAT_UINT16,//无符号 16 位数据
        GRAY_FORMAT_FLOAT,//浮点型数据
        GRAY_FORMAT_BGRD,//每像素 32 位， 按B/G/R/D顺序存放

}GRAY_FORMAT;
```

描述：

灰度数据格式。

类型：

GRAY_FORMAT_UINT8	8 位灰度数据
GRAY_FORMAT_UINT16	无符号 16 位灰度数据
GRAY_FORMAT_FLOAT	浮点型数据灰度
GRAY_FORMAT_BGRD	32 位灰度数据，每像素 32 位， 按 B/G/R/D 顺序存放

3.26 PointData

原型：

```
typedef struct tagPointData
{
   FLOAT32      x;
   FLOAT32      y;
   FLOAT32      z;
}PointData;
```

描述：

TOF 点云的数据结构.

参数：

x	点云 X 坐标值
y	点云 Y 坐标值
z	点云 Z 坐标值

3.27 RgbDData

原型：

```
typedef struct tagRgbDData
{
	UINT8 b;
	UINT8 g;
	UINT8 r;
}RgbDData;
```

描述：

TOF 设备 RGBD 的数据结构

参数：

b	颜色的蓝色分量
g	颜色的绿色分量
r	颜色的红色分量

3.28 COLOR_FORMAT

原型：

```
typedef enum tagCOLOR_FORMAT
{
	//MJPG格式
	COLOR_FORMAT_MJPG   = MAKE_UNIQUE_ID('M', 'J', 'P', 'G'),

	//H264 格式
	COLOR_FORMAT_H264   = MAKE_UNIQUE_ID('H', '2', '6', '4'),

	//YUV格式
	COLOR_FORMAT_YUV422 = MAKE_UNIQUE_ID('Y', 'U', 'V', 0x22),
	COLOR_FORMAT_YUYV   = MAKE_UNIQUE_ID('Y', 'U', 'Y', 'V'),
	COLOR_FORMAT_I420   = MAKE_UNIQUE_ID('I', '4', '2', '0'),
	COLOR_FORMAT_YV12   = MAKE_UNIQUE_ID('Y', 'V', '1', '2'),
	COLOR_FORMAT_NV12   = MAKE_UNIQUE_ID('N', 'V', '1', '2'),
	COLOR_FORMAT_NV21   = MAKE_UNIQUE_ID('N', 'V', '2', '1'),

	//RGB格式
	COLOR_FORMAT_BGR    = MAKE_UNIQUE_ID('B', 'G', 'R', 0x00), //RGB24（每个像素占 3 个字节，按照B、G、R的顺序存放）
```

COLOR_FORMAT_RGB = MAKE_UNIQUE_ID('R', 'G', 'B', 0x00), //RGB24（每个像素占 3 个字节，按照R、G、B的顺序存放）
COLOR_FORMAT_BGRA = MAKE_UNIQUE_ID('B', 'G', 'R', 'A'), //RGB32（每个像素占 4 个字节，按照B、G、R、A的顺序存放）
COLOR_FORMAT_RGBA = MAKE_UNIQUE_ID('R', 'G', 'B', 'A'), //RGB32（每个像素占 4 个字节，按照R、G、B、A的顺序存放）
}COLOR_FORMAT;

描述：

RGB 数据格式类型。

类型：

COLOR_FORMAT_MJPG	MJPG 格式
COLOR_FORMAT_H264	H264 格式
COLOR_FORMAT_YUV422	YUV422 格式
COLOR_FORMAT_YUYV	YUYV 格式
COLOR_FORMAT_I420	I420 格式
COLOR_FORMAT_YV12	YV12 格式
COLOR_FORMAT_NV12	NV12 格式
COLOR_FORMAT_NV21	NV21 格式
COLOR_FORMAT_BGR	RGB24（每个像素占 3 个字节，按照 B、G、R 的顺序存放）
COLOR_FORMAT_RGB	RGB24（每个像素占 3 个字节，按照 R、G、B 的顺序存放）
COLOR_FORMAT_BGRA	RGB32（每个像素占 4 个字节，按照 B、G、R、A 的顺序存放）
COLOR_FORMAT_RGBA	RGB32（每个像素占 4 个字节，按照 R、G、B、A 的顺序存放）

3.29 RgbData

原型：

```
typedef struct tagRgbData
{
   UINT8       r;
   UINT8       g;
   UINT8       b;
}RgbData;
```

描述：

TOF 设备 RGB 的数据结构;

参数：

r	颜色的红色分量
g	颜色的绿色分量
b	颜色的蓝色分量

3.30 RgbModuleLensParameter

原型：

```
typedef struct tagRgbModuleLensParameter
{
        FLOAT32 fx;
        FLOAT32 fy;
        FLOAT32 cx;
        FLOAT32 cy;
        FLOAT32 k1;
        FLOAT32 k2;
        FLOAT32 p1;
        FLOAT32 p2;
        FLOAT32 k3;
        //FLOAT32 k4;
}RgbModuleLensParameter;
```

描述：

RGB 模组内参和畸变。

类型：

fx	
fy	
cx	
cy	
k1	
k2	
p1	
p2	
k3	

3.31 StereoLensParameter

原型：

```
typedef struct tagStereoLensParameter
{
        FLOAT32 szRotationMatrix[3][3];//_双目旋转矩阵
        FLOAT32 szTranslationMatrix[3];//_双目平移矩阵
}StereoLensParameter;
```

描述：

双目相机参数。

类型：

szRotationMatrix	双目旋转矩阵
szTranslationMatrix	双目平移矩阵

3.32 TofExpouse

原型：

```
typedef struct tagTofExpouse
{
        UINT32          nCurrent;//当前值，可读写
        UINT32          nDefault;//默认值，只读
        UINT32          nStep;//步进值，只读
        UINT32          nMax;//最大值，只读
        UINT32          nMin;//最小值，只读
}TofExpouse;
```

描述：

TOF 曝光参数。

类型：

nCurrent	当前值
nDefault	默认值
nStep	步进值
nMax	最大值
nMin	最小值

3.33 TofExpouseGroup1

原型：

```
typedef struct tagTofExpouseGroup1
{
        TofExpouse exp;//曝光参数
}TofExpouseGroup1;
```

描述：

TOF 曝光参数（组合方式 1）。

类型：

exp	曝光参数

3.34 TofExpouseGroup2

原型：

```
typedef struct tagTofExpouseGroup2
{
        TofExpouse exp_AEF;//自动曝光帧曝光参数
        TofExpouse exp_FEF;//固定曝光帧曝光参数
}TofExpouseGroup2;
```

描述：

TOF 曝光参数（组合方式 2）。

类型：

exp_AEF	自动曝光帧曝光参数
exp_FEF	固定曝光帧曝光参数

3.35 TofExpouseGroup3

原型：

```
typedef struct tagTofExpouseGroup3
{
        TofExpouse exp_AEF;//自动曝光帧曝光参数
        TofExpouse exp_FEF;//固定曝光帧曝光参数
```

```
    TofExpouse exp_Gray;//灰度曝光帧曝光参数
}TofExpouseGroup3;
```

描述：

TOF 曝光参数（组合方式 3）。

类型：

exp_AEF	自动曝光帧曝光参数
exp_FEF	固定曝光帧曝光参数
exp_Gray	灰度曝光帧曝光参数

3.36 TofExpouseItems

原型：

```
typedef struct tagTofExpouseItems
{
    UINT32 nIndex;//1---g1 有效, 2---g2 有效, 3---g3 有效

    union
    {
        //[第 1 种]: 仅适用于只有单频或者双频raw数据的时候
        TofExpouseGroup1 g1;//曝光参数

        //[第 2 种]: 仅适用于具有自动曝光帧和固定曝光帧的raw数据的时候（帧内HDRZ融合时）
        TofExpouseGroup2 g2;//曝光参数

        //[第 3 种]: 仅适用于具有自动曝光帧和固定曝光帧的raw数据的时候（帧内HDRZ融合时），并且还可以配置灰度曝光帧的曝光
        TofExpouseGroup3 g3;//曝光参数
    }uParam;

}TofExpouseItems;
```

描述：

TOF 曝光参数选项组合。

类型：

nIndex	1---g1 有效, 2---g2 有效, 3---g3 有效
g1	曝光参数
g2	曝光参数

g3	曝光参数

3.37 TofExpouseCurrentGroup1

原型：

```
typedef struct tagTofExpouseCurrentGroup1
{
        UINT32 exp;//曝光值
}TofExpouseCurrentGroup1;
```

描述：

TOF 曝光值（组合方式 1）。

类型：

exp	曝光值

3.38 TofExpouseCurrentGroup2

原型：

```
typedef struct tagTofExpouseCurrentGroup2
{
        UINT32 exp_AEF;//自动曝光帧曝光值
        UINT32 exp_FEF;//固定曝光帧曝光值
}TofExpouseCurrentGroup2;
```

描述：

TOF 曝光值（组合方式 2）。

类型：

exp_AEF	自动曝光帧曝光值
exp_FEF	固定曝光帧曝光值

3.39 TofExpouseCurrentGroup3

原型：

```
typedef struct tagTofExpouseCurrentGroup3
{
        UINT32 exp_AEF;//自动曝光帧曝光值
        UINT32 exp_FEF;//固定曝光帧曝光值
        UINT32 exp_Gray;//灰度曝光帧曝光值
}TofExpouseCurrentGroup3;
```

描述：

TOF 曝光值（组合方式 3）。

类型：

exp_AEF	自动曝光帧曝光值
exp_FEF	固定曝光帧曝光值
exp_Gray	灰度曝光帧曝光值

3.40 TofExpouseCurrentItems

原型：

```
typedef struct tagTofExpouseCurrentItems
{
        UINT32 nIndex;//1---g1 有效, 2---g2 有效, 3---g3 有效

        union
        {
                //[第 1 种]: 仅适用于只有单频或者双频raw数据的时候
                TofExpouseCurrentGroup1 g1;//曝光值

                //[第 2 种]: 仅适用于具有自动曝光帧和固定曝光帧的raw数据的时候（帧内HDRZ融合时）
                TofExpouseCurrentGroup2 g2;//曝光值

                //[第 3 种]: 仅适用于具有自动曝光帧和固定曝光帧的raw数据的时候（帧内HDRZ融合时），并且还可以配置灰度曝光帧的曝光
                TofExpouseCurrentGroup3 g3;//曝光值
        }uParam;

}TofExpouseCurrentItems;
```

描述：

TOF 曝光值选项组合。

类型：

nIndex	1---g1 有效, 2---g2 有效, 3---g3 有效
g1	曝光值
g2	曝光值
g3	曝光值

3.41 TofExpouseRangeGroup1

原型：

```
typedef struct tagTofExpouseRangeGroup1
{
        UINT32 min;//曝光值(最小)
        UINT32 max;//曝光值(最大)
}TofExpouseRangeGroup1;
```

描述：

TOF 曝光范围（组合方式 1）。

类型：

min	曝光值(最小)
max	曝光值(最大)

3.42 TofExpouseRangeGroup2

原型：

```
typedef struct tagTofExpouseRangeGroup2
{
        UINT32 min_AEF;//自动曝光帧曝光值(最小)
        UINT32 max_AEF;//自动曝光帧曝光值(最大)

        UINT32 min_FEF;//固定曝光帧曝光值(最小)
        UINT32 max_FEF;//固定曝光帧曝光值(最大)
}TofExpouseRangeGroup2;
```

描述：

TOF 曝光范围（组合方式 2）。

类型：

min_AEF	自动曝光帧曝光值(最小)
max_AEF	自动曝光帧曝光值(最大)
min_FEF	固定曝光帧曝光值(最小)
max_FEF	固定曝光帧曝光值(最大)

3.43 TofExpouseRangeGroup3

原型：

```
typedef struct tagTofExpouseRangeGroup3
{
        UINT32 min_AEF;//自动曝光帧曝光值(最小)
        UINT32 max_AEF;//自动曝光帧曝光值(最大)

        UINT32 min_FEF;//固定曝光帧曝光值(最小)
        UINT32 max_FEF;//固定曝光帧曝光值(最大)

        UINT32 min_Gray;//灰度曝光帧曝光值(最小)
        UINT32 max_Gray;//灰度曝光帧曝光值(最大)
}TofExpouseRangeGroup3;
```

描述：

TOF 曝光范围（组合方式 3）。

类型：

min_AEF	自动曝光帧曝光值(最小)
max_AEF	自动曝光帧曝光值(最大)
min_FEF	固定曝光帧曝光值(最小)
max_FEF	固定曝光帧曝光值(最大)
min_Gray	灰度曝光帧曝光值(最小)
max_Gray	灰度曝光帧曝光值(最大)

3.44 TofExpouseRangeItems

原型：

```
typedef struct tagTofExpouseRangeItems
{
        UINT32 nIndex;//1---g1 有效, 2---g2 有效, 3---g3 有效

        union
        {
                //[第 1 种]: 仅适用于只有单频或者双频raw数据的时候
                TofExpouseRangeGroup1 g1;//曝光范围

                //[第 2 种]: 仅适用于具有自动曝光帧和固定曝光帧的raw数据的时候（帧内HDRZ融合时）
                TofExpouseRangeGroup2 g2;//曝光范围

                //[第 3 种]: 仅适用于具有自动曝光帧和固定曝光帧的raw数据的时候（帧内HDRZ融合时），并且还可以配置灰度曝光帧的曝光
                TofExpouseRangeGroup3 g3;//曝光范围
        }uParam;

}TofExpouseRangeItems;
```

描述：

TOF 曝光范围选项组合。

类型：

nIndex	1---g1 有效, 2---g2 有效, 3---g3 有效
g1	曝光范围
g2	曝光范围
g3	曝光范围

3.45 CUSTOM_PARAM_GUEST_ID

原型：

```
typedef enum tagCUSTOM_PARAM_GUEST_ID
{
        CUSTOM_PARAM_GUEST_ID_1 = 1,//客户 1
        CUSTOM_PARAM_GUEST_ID_2 = 2,//客户 2
}CUSTOM_PARAM_GUEST_ID;
```

描述：

自定义参数的客户识别号。

类型：

CUSTOM_PARAM_GUEST_ID_1	客户 1
CUSTOM_PARAM_GUEST_ID_2	客户 2

3.46 CustomParamGuest1

原型：

```
typedef struct tagCustomParamGuest1
{
        SINT32 quantileThreshold;//AE 比例
        FLOAT32 referenceAmplitude;//参考幅度
        FLOAT32 amplitudeThreshold;//幅度阈值
        UINT8 szRes[496];//总长 508 字节，4 字节对齐，预留
}CustomParamGuest1;
```

描述：

客户 1 自定义的参数。

类型：

quantileThreshold	AE 比例
referenceAmplitude	参考幅度
amplitudeThreshold	幅度阈值
szRes	字节对齐，预留

3.47 CustomParamGuest2

原型：

```
typedef struct tagCustomParamGuest2
{
        UINT8 szRes[508];//总长 508 字节，4 字节对齐，预留
```

```
}CustomParamGuest2;
```

描述：

客户 2 自定义的参数。

类型：

szRes	字节对齐，预留

3.48 GuestCustomParam

原型：

```
typedef struct tagGuestCustomParam
{
        CUSTOM_PARAM_GUEST_ID id;//输入参数，只读

        union
        {
                CustomParamGuest1 p1;//当id为CUSTOM_PARAM_GUEST_ID_1 时有效；
                CustomParamGuest2 p2;//当id为CUSTOM_PARAM_GUEST_ID_2 时有效；


                UINT8 data[508];//限定联合体为 508 字节长度（该字段不使用，仅用于数
据结构长度定义）
        }uParam;
}GuestCustomParam;
```

描述：

客户自定义的参数结构体。

类型：

id	自定义参数的客户 ID	
uParam	根据 id 参数（客户 id）的不同，使用不同的字段；	
	p1	当 id 为 CUSTOM_PARAM_GUEST_ID_1 时有效
	p2	当 id 为 CUSTOM_PARAM_GUEST_ID_2 时有效
	data	限定联合体长度为 508 字节

3.49 RoiItem

原型：

```
typedef struct tagRoiItem
{
        UINT32  left;//起始列，从 0 开始;
        UINT32  top;//起始行，从 0 开始;
        UINT32  right;//终止列，不超过图像宽;
```

```
    UINT32  bottom;//终止行，不超过图像高;

}RoiItem;
```

描述：

ROI 区域结构体。

参数：

left	起始列，从 0 开始
top	起始行，从 0 开始
right	终止列，不超过图像宽
bottom	终止行，不超过图像高

3.50 DepthCalRoi

原型：

```
typedef struct tagDepthCalRoi
{
    RoiItem struMax;//最大值，只读
    RoiItem struDefault;//默认值，只读

    RoiItem struCurrent;//当前值，可读写

}DepthCalRoi;
```

描述：

ROI 区域结构体。

参数：

struMax	最大值，只读
struDefault	默认值，只读
struCurrent	当前值，可读写

3.51 TofModuleLensGeneral

原型：

```
typedef struct tagTofModuleLensGeneral
{
    FLOAT32 fx;
    FLOAT32 fy;
    FLOAT32 cx;
    FLOAT32 cy;
    FLOAT32 k1;
```

```
        FLOAT32 k2;
        FLOAT32 p1;
        FLOAT32 p2;
        FLOAT32 k3;
}TofModuleLensGeneral;
```

描述：

TOF 模组内参和畸变（通用模型）。

类型：

fx	
fy	
cx	
cy	
k1	
k2	
p1	
p2	
k3	

3.52 TofModuleLensFishEye

原型：

```
typedef struct tagTofModuleLensFishEye
{
        FLOAT32 fx;
        FLOAT32 fy;
        FLOAT32 cx;
        FLOAT32 cy;
        FLOAT32 k1;
        FLOAT32 k2;
        FLOAT32 k3;
        FLOAT32 k4;
}TofModuleLensFishEye;
```

描述：

TOF 模组内参和畸变（鱼眼模型）。

类型：

fx	
fy	
cx	

cy	
k1	
k2	
k3	
k4	

3.53 TofModuleLensParameter

原型：

```
typedef struct tagTofModuleLensParameter
{
        FLOAT32 fx;
        FLOAT32 fy;
        FLOAT32 cx;
        FLOAT32 cy;
        FLOAT32 k1;
        FLOAT32 k2;
        FLOAT32 p1;
        FLOAT32 p2;
        FLOAT32 k3;
        //FLOAT32 k4;
}TofModuleLensParameter;
```

描述：

TOF 模组内参和畸变（V1.0 版本，建议不要再用，因为不能适用于鱼眼模型）。

类型：

fx	
fy	
cx	
cy	
k1	
k2	
p1	
p2	
k3	

3.54 TofModuleLensParameterV20

原型：

```
typedef struct tagTofModuleLensParameterV20
{
	UINT32 nIndex;//1---general有效, 2---fishEye有效

	union
	{
		//[第 1 种]: 普通模型
		TofModuleLensGeneral general;//普通模型

		//[第 2 种]: 鱼眼模型
		TofModuleLensFishEye fishEye;//鱼眼模型
	}uParam;

}TofModuleLensParameterV20;
```

描述：

TOF 模组内参和畸变（V2.0 版本）。

类型：

nIndex		1---general 有效, 2---fishEye 有效
uParam	general	普通模型
	fishEye	鱼眼模型

3.55 TofCalibData

原型：

```
typedef struct tagTofCalibData
{
	UINT8* pData;//指向标定数据
	UINT32 nDataLen;//pData内标定数据长度
}TofCalibData;
```

描述：

TOF 模组标定数据结构体。

类型：

pData	指向标定数据
nDataLen	pData 内标定数据长度

3.56 TofRawData

原型：

```
typedef struct tagTofRawData
{
        //RAW数据
        UINT8* pRaw;//一帧RAW数据
        UINT32 nRawLen;//RAW数据长度（字节数）

        //RAW数据其他属性参数
        FLOAT32 fTemperature;//出RAW数据时模组温度（注意：部分型号模组不需要该字段、部分模组RAW数据自带该数据，那么可以输入 0 值）

}TofRawData;
```

描述：

RAW 数据结构体。

类型：

pRaw	一帧 RAW 数据
nRawLen	RAW 数据长度**（字节数）**
fTemperature	出 RAW 数据时模组温度**（注意：部分型号模组不需要该字段、部分模组 RAW 数据自带该数据，那么可以输入 0 值）**

3.57 ExterntionHooks

原型：

```
typedef struct tagExterntionHooks
{
        void* pUserData;//用户自定义数据

        /**********用于提前送出计算出来的曝光值**********/
        //@    pExp:  计算出的曝光值信息;
        //@    user_data:  用户自定义数据，与pUserData属于同一个;
        //@    【特别注意】: 对于在该回调函数内调用TOFM_XXX接口时，只允许调用软件算法部分接口，否则会死锁！！！！！
        void(*RecvTofExpTime)(TofExpouseCurrentItems* pExp, void*user_data);//根据模组实际情况选择是否实现


}ExterntionHooks;
```

描述：

RAW 数据结构体。

类型：

pUserData	用户自定义数据
RecvTofExpTime	用于提前送出计算出来的曝光值(针对同一时间只出单频或者支出双频 raw 数据的模式时有效)； 【特别注意】： 对于在该回调函数内调用 TOFM_XXX 接口时，只允许调用软件算法部分接口，否则会死锁！！！！！ 【关于参数的说明：】 pExp：计算出的曝光值； user_data: 用户自定义数据，与 pUserData 属于同一个；

4 特有数据结构与类型定义

4.1 TOF_DEV_TYPE

原型：

```
typedef enum tagTOF_DEV_TYPE
{
	TOF_DEV_CHROMEBOOK                  = MAKE_UNIQUE_ID('C',  'M',  'B',
0x00),//ChromeBook
	TOF_DEV_CLEANER01A        = MAKE_UNIQUE_ID('C', 0x01,  'A',
0x00),//Cleaner01A
	TOF_DEV_CLEANER01APLUS     = MAKE_UNIQUE_ID('C', 0x01,  'A',
0x01),//Cleaner01A（Plus版）
	TOF_DEV_CLEANER01APRO      = MAKE_UNIQUE_ID('C', 0x01,  'A',
0x02),//Cleaner01A（Pro版）
	TOF_DEV_CLEANER01A_NET     = MAKE_UNIQUE_ID('C', 0x01,  'A',
0x02),//Cleaner01A（网络版）
	TOF_DEV_CLEANER01B        = MAKE_UNIQUE_ID('C', 0x01,  'B',
0x00),//Cleaner01B
	TOF_DEV_CLEANER01D        = MAKE_UNIQUE_ID('C', 0x01,  'D',
0x00),//Cleaner01D
	TOF_DEV_CLEANER01D_NET     = MAKE_UNIQUE_ID('C', 0x01,  'D',
0x01),//Cleaner01D（网络版）
	TOF_DEV_CLEANER01E_NET     = MAKE_UNIQUE_ID('C', 0x01,  'E',
0x01),//Cleaner01E（网络版）
	TOF_DEV_CLEANER01F        = MAKE_UNIQUE_ID('C', 0x01,  'F',
0x00),//Cleaner01F
	TOF_DEV_CLEANER01F1       = MAKE_UNIQUE_ID('C', 0x01,  'F',
0x01),//Cleaner01F1
	TOF_DEV_CLEANER01G        = MAKE_UNIQUE_ID('C', 0x01,  'G',
0x00),//Cleaner01G
	TOF_DEV_CLEANER01G1       = MAKE_UNIQUE_ID('C', 0x01,  'G',
0x01),//Cleaner01G1
	TOF_DEV_CLEANER01X        = MAKE_UNIQUE_ID('C', 0x01,  'X',
0x00),//Cleaner01X
	TOF_DEV_CLEANER02A        = MAKE_UNIQUE_ID('C', 0x02,  'A',
0x00),//Cleaner02A
	TOF_DEV_CLEANER02A_NET     = MAKE_UNIQUE_ID('C', 0x02,  'A',
0x01),//Cleaner02A（网络版）
	TOF_DEV_MARS01A          = MAKE_UNIQUE_ID('M', 0x01,  'A', 0x00),//Mars01A
	TOF_DEV_MARS01B          = MAKE_UNIQUE_ID('M', 0x01,  'B', 0x00),//Mars01B
	TOF_DEV_MARS01C          = MAKE_UNIQUE_ID('M', 0x01,  'C', 0x00),//Mars01C
	TOF_DEV_MARS01D          = MAKE_UNIQUE_ID('M', 0x01,  'D', 0x00),//Mars01D
	TOF_DEV_MARS01E          = MAKE_UNIQUE_ID('M', 0x01,  'E', 0x00),//Mars01E
	TOF_DEV_MARS01H          = MAKE_UNIQUE_ID('M', 0x01,  'H', 0x00),//Mars01H
	TOF_DEV_MARS04           = MAKE_UNIQUE_ID('M', 0x04, 0x00, 0x00),//Mars04
	TOF_DEV_MARS04A          = MAKE_UNIQUE_ID('M', 0x04,  'A', 0x00),//Mars04A
	TOF_DEV_MARS04B          = MAKE_UNIQUE_ID('M', 0x04,  'B', 0x00),//Mars04B
	TOF_DEV_MARS05           = MAKE_UNIQUE_ID('M', 0x05, 0x00, 0x00),//Mars05
	TOF_DEV_MARS05A          = MAKE_UNIQUE_ID('M', 0x05,  'A', 0x00),//Mars05A
	TOF_DEV_MARS05B          = MAKE_UNIQUE_ID('M', 0x05,  'B', 0x00),//Mars05B
	TOF_DEV_MARS05B_BCTC       = MAKE_UNIQUE_ID('M', 0x05,  'B',
0x01),//Mars05B(BCTC版本)
```

```
        TOF_DEV_MARS05B_BCTC_SUNNY = MAKE_UNIQUE_ID('M', 0x05,  'B',
0x02),//Mars05B(BCTC版本_sunny)
        TOF_DEV_USBTOF_HI        = MAKE_UNIQUE_ID('U',  'T',  'H', 0x00),//UsbTof-Hi
        TOF_DEV_DREAM            = MAKE_UNIQUE_ID('D',  'R',  'M', 0x00),//DREAM
        TOF_DEV_HOT002           = MAKE_UNIQUE_ID('H',  'O',  'T', 0x02),//HOT002
        TOF_DEV_HOT002A          = MAKE_UNIQUE_ID('H',  'O',  'T', 0x2a),//HOT002A
        TOF_DEV_SEEKER07C                = MAKE_UNIQUE_ID('S',  'E',  'K',
0x7C),//seeker07c
        TOF_DEV_SEEKER08A        = MAKE_UNIQUE_ID('S',  'E',  'K', 0x8A),//seeker08A
        TOF_DEV_LOGITECH_C525    = MAKE_UNIQUE_ID('L',  'G', 0xC5,
0x25),//Logitech C525


        //这部分为demo模块
        TOF_DEV_DEMO_3DCP_NET    = MAKE_UNIQUE_ID(0xde, 0x3d, 'C',
0x00),//demo版 3DCP（网络版）
        TOF_DEV_DEMO_3DCP        = MAKE_UNIQUE_ID(0xde, 0x3d, 'C', 0x01),//demo版
3DCP
        TOF_DEV_DEMO_C00P01A_NET  = MAKE_UNIQUE_ID(0xde, 0xC0, 'P',
0x1A),//demo版C00P01A的RGBD模块（网络版）
        TOF_DEV_DEMO_UPG         = MAKE_UNIQUE_ID(0xde,  'U',  'P',  'G'),//demo版
UPG

}TOF_DEV_TYPE;
```

描述：

TOF 设备型号。

类型：

TOF_DEV_CHROMEBOOK	ChromeBook
TOF_DEV_CLEANER01A	Cleaner01A
TOF_DEV_CLEANER01APLUS	Cleaner01A（Plus 版）
TOF_DEV_CLEANER01APRO	Cleaner01A（Pro 版）
TOF_DEV_CLEANER01A_NET	Cleaner01A（网络版）
TOF_DEV_CLEANER01B	Cleaner01B
TOF_DEV_CLEANER01D	Cleaner01D
TOF_DEV_CLEANER01D_NET	Cleaner01D（网络版）
TOF_DEV_CLEANER01E_NET	Cleaner01E（网络版）
TOF_DEV_CLEANER01F	Cleaner01F
TOF_DEV_CLEANER01F1	Cleaner01F1
TOF_DEV_CLEANER01G	Cleaner01G
TOF_DEV_CLEANER01G1	Cleaner01G1
TOF_DEV_CLEANER01X	Cleaner01X
TOF_DEV_CLEANER02A	Cleaner02A

TOF_DEV_CLEANER02A_NET	Cleaner02A（网络版）
TOF_DEV_MARS01A	Mars01A
TOF_DEV_MARS01B	Mars01B
TOF_DEV_MARS01C	Mars01C
TOF_DEV_MARS01D	Mars01D
TOF_DEV_MARS01E	Mars01E
TOF_DEV_MARS01H	Mars01H
TOF_DEV_MARS04	Mars04
TOF_DEV_MARS04A	Mars04A
TOF_DEV_MARS04B	Mars04B
TOF_DEV_MARS05	Mars05
TOF_DEV_MARS05A	Mars05A
TOF_DEV_MARS05B	Mars05B
TOF_DEV_MARS05B_BCTC	Mars05B(BCTC 版本)
TOF_DEV_MARS05B_BCTC_SUNNY	Mars05B(BCTC 版本_sunny)
TOF_DEV_USBTOF_HI	UsbTof-Hi
TOF_DEV_DREAM	DREAM
TOF_DEV_HOT002	HOT002
TOF_DEV_HOT002A	HOT002A
TOF_DEV_SEEKER07C	SEEKER07C
TOF_DEV_SEEKER08A	SEEKER08A
TOF_DEV_LOGITECH_C525	Logitech C525
TOF_DEV_DEMO_3DCP_NET	demo 版 3DCP（网络版）
TOF_DEV_DEMO_3DCP	demo 版 3DCP
TOF_DEV_DEMO_C00P01A_NET	demo 版 C00P01A 的 RGBD 模块（网络版）
TOF_DEV_DEMO_UPG	demo 版 UPG

4.2 TofFrameData

原型：

```
typedef struct tagTofFrameData
{
        UINT64  timeStamp;
        UINT32  frameWidth;
        UINT32  frameHeight;
```

```
//
FLOAT32* pDepthData;//射线距离（滤波前）
FLOAT32* pDepthDataFilter;//射线距离（滤波后）

PointData *pPointData;//点云数据

GRAY_FORMAT grayFormat;//pGrayData内数据格式
void   *pGrayData;//灰度数据

FLOAT32 *pConfidence;//置信度（支持的板子才可以）

RgbDData* pRgbD;//RgbD数据

void   *pRawData;//raw数据（支持raw数据的板子才可以）
UINT32 nRawDataLen;//pRawData内raw数据长度，字节数

//扩展数据(一般针对客户特殊需求)，不同设备/不同客户均不同，可能为空；
void   *pExtData;//扩展数据
UINT32 nExtDataLen;//pExtData内扩展数据长度，字节数
```

}TofFrameData;

描述：

TOF 数据结构体.

参数：

timeStamp	TOF 数据帧的时间戳。
frameWidth	TOF 数据帧宽度
frameHeight	TOF 数据帧高度
pDepthData	射线距离（滤波前）
pDepthDataFilter	射线距离（滤波后）
pPointData	TOF 点云数据
grayFormat	pGrayData 内数据格式
pGrayData	TOF IR 图像数据
pConfidence	置信度（支持的板子才可以）
pRgbD	RGBD 数据（rgbd 支持情况下）
pExtData	扩展数据(一般针对客户特殊需求)，不同设备/不同客户均不同，可能为空。
nExtDataLen	pExtData 内扩展数据长度，字节数

4.3 ANALOG_GAIN_MODE

原型：

```
typedef enum tagANALOG_GAIN_MODE
{
        ANALOG_GAIN_MODE_MANUAL = 0x00000001,//手动模拟增益
        ANALOG_GAIN_MODE_AUTO   = 0x00000002,//自动模拟增益
}ANALOG_GAIN_MODE;
```

描述：

TOF 模拟增益类型。

类型：

ANALOG_GAIN_MODE_MANUAL	手动模拟增益
ANALOG_GAIN_MODE_AUTO	自动模拟增益

4.4 DIGITAL_GAIN_MODE

原型：

```
typedef enum tagDIGITAL_GAIN_MODE
{
        DIGITAL_GAIN_MODE_MANUAL = 0x00000001,//手动数字增益
        DIGITAL_GAIN_MODE_AUTO   = 0x00000002,//自动数字增益
}DIGITAL_GAIN_MODE;
```

描述：

TOF 数字增益类型。

类型：

DIGITAL_GAIN_MODE_MANUAL	手动数字增益
DIGITAL_GAIN_MODE_AUTO	自动数字增益

4.5 RgbVideoControlProperty

原型：

```
typedef enum tagRgbVideoControlProperty
{
        RgbVideoControl_Exposure = 0x00000001,//RGB模组的曝光属性
        RgbVideoControl_Gain     = 0x00000002,//RGB模组的增益属性
}RgbVideoControlProperty;
```

描述：

RGB 属性类型。

类型：

RgbVideoControl_Exposure	曝光属性
RgbVideoControl_Gain	增益属性

4.6 RgbVideoControlFlags

原型：

```
typedef enum tagRgbVideoControlFlags
{
        RgbVideoControlFlags_Auto   = 0x00000001,//自动
        RgbVideoControlFlags_Manual = 0x00000002,//手动
}RgbVideoControlFlags;
```

描述：

RGB 属性值（扩展属性值）。

类型：

RgbVideoControlFlags_Auto	自动
RgbVideoControlFlags_Manual	手动

4.7 RgbVideoControl

原型：

```
typedef struct tagRgbVideoControl
{
        SINT32          lDefault;//默认值
        SINT32          lStep;//步进值
        SINT32          lMax;//最大值
        SINT32          lMin;//最小值
        SINT32    lCapsFlags;//支持的值，是RgbVideoControlFlags的一种或多种组合

        SINT32          lCurrent;//当前值
        RgbVideoControlFlags    lFlags;//当前Flag值

}RgbVideoControl;
```

描述：

RGB 属性值。

类型：

lDefault	默认值
lStep	步进值
lMax	最大值
lMin	最小值
lCapsFlags	支持的值，是 RgbVideoControlFlags 的一种或多种组合
lCurrent	当前值
lFlags	当前 Flag 值，是 RgbVideoControlFlags 的一种

4.8 RgbFrameData

原型：

```
typedef struct tagRgbFrameData
{
        UINT64  timeStamp;
        UINT32  frameWidth;
        UINT32  frameHeight;

        COLOR_FORMAT formatType;//指明pFrameData内数据帧的格式
        COLOR_FORMAT formatTypeOrg;//指明pFrameData内数据帧的格式(编码压缩之前的格式)
        UINT32  nFrameLen;
        UINT8*  pFrameData;

        //扩展数据(一般针对客户特殊需求)，不同设备/不同客户均不同，可能为空；
        void   *pExtData;//扩展数据
        UINT32 nExtDataLen;//pExtData内扩展数据长度，字节数

}RgbFrameData;
```

描述：

RGB 数据结构体.

参数：

timeStamp	RGB 数据帧的时间戳。
frameWidth	RGB 数据帧宽度
frameHeight	RGB 数据帧高度
formatType	指明 pFrameData 内数据帧的格式
formatTypeOrg	指明 pFrameData 内数据帧的格式(编码压缩之前的格式)
nFrameLen	指明 pFrameData 内数据帧的长度
pFrameData	RGB 数据

pExtData	扩展数据(一般针对客户特殊需求)，不同设备/不同客户均不同，可能为空。
nExtDataLen	pExtData 内扩展数据长度，字节数

4.9 ImuFrameData

原型：

```
typedef struct tagImuFrameData
{
        UINT64 timeStamp;

        FLOAT32 accelData_x;
        FLOAT32 accelData_y;
        FLOAT32 accelData_z;

        FLOAT32 gyrData_x;
        FLOAT32 gyrData_y;
        FLOAT32 gyrData_z;

        FLOAT32 magData_x;
        FLOAT32 magData_y;
        FLOAT32 magData_z;
}ImuFrameData;
```

描述：

IMU 数据结构体.

参数：

timsstamp	IMU 时间戳
accelData_x	x 轴向加速度
accelData_y	y 轴向加速度
accelData_z	z 轴向加速度
gyrData_x	x 轴向加速度
gyrData_y	x 轴向加速度
gyrData_z	x 轴向加速度
magData_x	x 轴向加速度
magData_y	x 轴向加速度
magData_z	x 轴向加速度

4.10 TofDevInitParam

原型：

```
typedef struct tagTofDevInitParam
```

{

SCHAR szDepthCalcCfgFileDir[200]; //深度计算所需配置文件的目录，如 home/user/temp

UINT8 nLogLevel; //日志打印级别（暂时还未生效）

SBOOL bSupUsb; //是否支持USB设备

SBOOL bSupNetWork; //是否支持网络设备

SCHAR szHostIPAddr[32]; //本地主机上的某一个网卡的IP地址（也可不填，不填的情况下将会遍历本地所有网卡）

SBOOL bSupSerialCOM; //是否需要支持串口（需要使用串口时才需赋值true）

SCHAR szSerialDev[64]; //本地主机上的某一个串口设备（当bSupSerialCOM字段为true时，该字段才有效）

//windows环境下可以不填写，也可以填写，如COM1、COM2、...，不填写的情况下将会遍历本地所有串口

//linux环境下必须填写，如/dev/ttyS0、/dev/ttyUSB0、...

SBOOL bWeakAuthority;//是否是权限较低（如非ROOT的安卓系统）,仅适用于linux系统/安卓系统

SBOOL bDisablePixelOffset;//SDK在内部不进行地址偏移后输出（输出给用户的TOF数据分辨率与RAW数据相同）

SCHAR szLogFile[256]; //SDK内部记录debug信息的日志文件，如 home/user/temp/tof_dev_sdk_log.txt

}TofDevInitParam;

描述：

TOF 模块 SDK 的初始化参数结构体.

参数：

szDepthCalcCfgFileDir	深度计算所需配置文件的目录，如 home/user/temp
nLogLevel	日志打印级别（暂时还未生效）
bSupUsb	是否支持 USB 设备
bSupNetWork	是否支持网络设备
szHostIPAddr	本地主机上的某一个网卡的 IP 地址（也可不填，不填的情况下将会遍历本地所有网卡）
bSupSerialCOM	是否需要支持串口（需要使用串口时才需赋值 true）
szSerialDev	本地主机上的某一个串口设备（当 bSupSerialCOM 字段为 true 时，该字段才有效）； windows 环境下可以不填写，也可以填写，如 COM1、COM2、...，不填写的情况下将会遍历本地所有串口； linux 环境下必须填写，如/dev/ttyS0、/dev/ttyUSB0、...；

bWeakAuthority	是否是权限较低（如非 ROOT 的安卓系统）,仅适用于 linux 系统/安卓系统
bDisablePixelOffset	SDK 在内部不进行地址偏移后输出 TOF 数据（输出给用户的 TOF 数据分辨率与 RAW 数据相同）
szLogFile	SDK 内部记录 debug 信息的日志文件，如 home/user/temp/tof_dev_sdk_log.txt

4.11 TofDeviceDescriptor

原型：

```
typedef struct tagTofDeviceDescriptor
{
        void* hDevice;
        void* hDriver;
}TofDeviceDescriptor;
```

描述：

TOF 设备描述结构体.

参数：

hDevice	TOF 设备句柄，SDK 内部使用
hDriver	TOF 设备驱动句柄，SDK 内部使用

4.12 TofDeviceDescriptorWithFd

原型：

```
typedef struct tagTofDeviceDescriptorWithFd
{
        SINT32 usbDevFd; //USB设备的描述符fd
        UINT16 usbDevVID;//USB设备的VID
        UINT16 usbDevPID;//USB设备的PID
}TofDeviceDescriptorWithFd;
```

描述：

TOF 设备描述结构体(具有设备句柄 fd).

参数：

usbDevFd	USB设备的描述符fd
usbDevVID	USB 设备的 VID
usbDevPID	USB 设备的 PID

4.13 TofDeviceInfo

原型：

```
typedef struct tagTofDeviceInfo
{
        //BASIC information
        TOF_DEV_TYPE devType;//用于区分是哪款设备
        SCHAR szDevName[32];
        SCHAR szDevId[64];//设备/模块的序列号（标识设备唯一性）
        SCHAR szFirmwareVersion[32];//固件版本信息

        //TOF
        UINT32 supportedTOFMode;//TOF_MODE的组合
        UINT32 tofResWidth;
        UINT32 tofResHeight;
        GRAY_FORMAT grayFormat;//灰度数据格式

        //TOF Expouse
        UINT32 supportedTofExpMode;//EXP_MODE的组合
        //TOF Analog Gain
        UINT32 supportedTofAnalogGainMode;//ANALOG_GAIN_MODE的组合
        //TOF Digital Gain
        UINT32 supportedTofDigitalGainMode;//DIGITAL_GAIN_MODE的组合

        //TOF Filter
        UINT32 supportedTOFFilter; //TOF_FILTER的组合

        //TOF HDRZ
        SBOOL bTofHDRZSupported;
        UINT8 byRes1[3];//字节对齐，预留

        //TOF RemoveINS
        SBOOL bTofRemoveINSSupported;
        UINT8 byRes5[3];//字节对齐，预留

        //TOF MPIFlag
        SBOOL bTofMPIFlagSupported;//[该字段已作废]
        UINT8 byRes6[3];//字节对齐，预留

        //RGB
        SBOOL bRgbSupported;
        UINT8 byRes2[3];//字节对齐，预留
        COLOR_FORMAT rgbColorFormat;//传出的RGB数据格式
        COLOR_FORMAT rgbColorFormatOrg;//传出的RGB数据格式(编码压缩之前的格式)
        UINT32 rgbResWidth;
        UINT32 rgbResHeight;
        UINT32 supportedRgbProperty;// RgbVideoControlProperty的组合

        //RGBD
        SBOOL bRgbDSupported;
        UINT8 byRes3[3];//字节对齐，预留

        //IMU
```

```
    SBOOL bImuSupported;
    UINT8 byRes4[3];//字节对齐，预留

    //远程抓图
    SBOOL bRemoteCaptureSupported;
    //固件升级
    SBOOL bUpgradeFirmwareSupported;
    //设备重启
    SBOOL bRebootDevSupported;
    //主从机间同步时间
    SBOOL bMasterSlaveSyncTimeSupported;

    //
}TofDeviceInfo;
```

描述：

TOF 设备信息数据结构体.

参数：

devType	用于区是哪款设备
szDevName	TOF 设备名称，用于区分模块种类
szDevId	TOF 设备/模块的序列号（标识设备唯一性）
szFirmwareVersion	TOF 设备固件版本信息
supportedTOFMode	TOF 设备支持的 TOF 模式，可以是 TOF_MODE 的各种组合
tofResWidth	TOF 分辨率宽度信息
tofResHeight	TOF 分辨率高度信息
grayFormat	灰度数据格式
supportedTofExpMode	TOF 设备支持的 TOF 曝光模式，可以是 EXP_MODE 的各种组合
supportedTofAnalogGainMode	TOF 设备支持的 TOF 模拟增益模式，可以是 ANALOG_GAIN_MODE 的组合
supportedTofDigitalGainMode	TOF 设备支持的 TOF 数字增益模式，可以是 DIGITAL_GAIN_MODE 的组合
supportedTOFFilter	TOF 设备支持的 TOF 滤波种类，可以是 TOF_FILTER 的各种组合
bTofHDRZSupported	TOF 设备是否支持 HDRZ 输出
bTofRemoveINSSupported	TOF 设备是否支持 RemoveINS 算法
bTofMPIFlagSupported	TOF 设备是否支持 MPIFlag 算法[该字段已作废]
bRgbSupported	TOF 设备是否支持 RGB 输出
rgbColorFormat	传出的 RGB 数据格式

rgbColorFormatOrg	传出的 RGB 数据格式(编码压缩之前的格式)
rgbResWidth	RGB 分辨率宽度信息
rgbResHeight	RGB 分辨率高度信息
supportedRgbProperty	RGB 支持的属性
bRgbDSupported	TOF 设备是否支持 RGBD 输出
bImuSupported	TOF 设备支持 IMU 输出
bRemoteCaptureSupported	TOF 设备支持远程抓图功能（存到设备内部）
bUpgradeFirmwareSupported	TOF 设备支持固件升级
bRebootDevSupported	TOF 设备支持重启设备

4.14 TofDeviceParam

原型：

```
typedef struct tagTofDeviceParam
{
        FLOAT32 fBoardTemp;//主板温度(需要设备支持)
        FLOAT32 fSensorTemp;//senseor温度(需要设备支持)
        FLOAT32 fImuTemp;//Imu温度(需要设备支持)
}TofDeviceParam;
```

描述：

设备参数（一般是一些动态变化的只读参数）。

类型：

fBoardTemp	主板温度(需要设备支持)，（0.0 一般表示不支持）
fSensorTemp	Sensor 温度(需要设备支持)，（0.0 一般表示不支持）
fImuTemp	Imu 温度(需要设备支持)，（0.0 一般表示不支持）

4.15 TofDeviceTemperature

原型：

```
typedef struct tagTofDeviceTemperature
{
        FLOAT32 fBoardTemp;//主板温度(需要设备支持)
        FLOAT32 fSensorTemp;//senseor温度(需要设备支持)
        FLOAT32 fImuTemp;//Imu温度(需要设备支持)
}TofDeviceTemperature;
```

描述：

设备温度信息参数（不同的设备获取的温度种类不同）。

类型：

fBoardTemp	主板温度(需要设备支持)，（0.0 一般表示不支持）
fSensorTemp	Sensor 温度(需要设备支持)，（0.0 一般表示不支持）
fImuTemp	Imu 温度(需要设备支持)，（0.0 一般表示不支持）

4.16 NetDevInfo_t

原型：

```
typedef struct tagNetDevInfo
{
        SBOOL bDHCP;//是否是自动获取IP
        UINT8 byRes[3];//字节对齐，预留
        SCHAR szIPv4Address[32];//设备IP地址
        SCHAR szIPv4SubnetMask[32];//设备子网掩码
        SCHAR szIPv4Gateway[32];//设备网关
        SCHAR szMAC[32];//设备MAC地址

}NetDevInfo_t;
```

描述：

设备的网络信息参数（网络接入方式的设备才支持）。

类型：

bDHCP	是否是自动获取 IP
szIPv4Address	设备 IP 地址
szIPv4SubnetMask	设备子网掩码
szIPv4Gateway	设备网关
szMAC	设备 MAC 地址

4.17 RemoteCapture

原型：

```
typedef struct tagRemoteCapture
{
        UINT8 szRes[4];//预留,4 字节对齐
}RemoteCapture;
```

描述：

远程控制设备抓图并保存到设备内部（部分设备支持）。

类型：

szRes	无实际意义，预留为字节对齐。

4.18 FIRMWARE_UPGRADE_STATUS

原型：

```
typedef enum tagFIRMWARE_UPGRADE_STATUS
{
        FIRMWARE_UPGRADE_STATUS_FINISHED   = 1,//升级完成
        FIRMWARE_UPGRADE_STATUS_RUNNING    = 2,//正在升级
        FIRMWARE_UPGRADE_STATUS_FAILED     = 3,//升级失败
        FIRMWARE_UPGRADE_STATUS_UNKNOWN    = 4,//升级失败(未知错误)
        FIRMWARE_UPGRADE_STATUS_ERROR_DATA = 5,//升级失败(固件包错误)
        FIRMWARE_UPGRADE_STATUS_IO         = 6,//升级失败(IO读写失败)
}FIRMWARE_UPGRADE_STATUS;
```

描述：

固件升级的实时状态。

类型：

FIRMWARE_UPGRADE_STATUS_FINISHED	升级完成
FIRMWARE_UPGRADE_STATUS_RUNNING	正在升级
FIRMWARE_UPGRADE_STATUS_FAILED	升级失败
FIRMWARE_UPGRADE_STATUS_UNKNOWN	升级失败(未知错误)
FIRMWARE_UPGRADE_STATUS_ERROR_DATA	升级失败(固件包错误)
FIRMWARE_UPGRADE_STATUS_IO	升级失败(IO 读写失败)

4.19 FirmwareUpgradeStatus

原型：

```
typedef struct tagFirmwareUpgradeStatus
{
        FIRMWARE_UPGRADE_STATUS status;//升级的状态，取值见
FIRMWARE_UPGRADE_STATUS
        UINT8 nProgress;//实时进度，取值必须处于：0-100
        UINT8 byRes[3];//字节对齐，预留
}FirmwareUpgradeStatus;
```

描述：

固件升级的实时状态信息。

类型：

status	升级的状态
nProgress	实时进度，取值必须处于：0-100
byRes	字节对齐，预留

4.20 FNFirmwareUpgradeStatus

原型：

typedef void (*FNFirmwareUpgradeStatus)(FirmwareUpgradeStatus *statusData, void* pUserData);

描述：

固件升级的实时状态回调函数。

参数：

statusData	固件升级的实时状态信息
pUserData	用户数据指针

4.21 FirmwareUpgradeData

原型：

```
typedef struct tagFirmwareUpgradeData
{
        UINT8* pData;//指向固件数据（完整的固件数据首地址）
        UINT32 nDataLen;//pData内固件数据长度（完整的固件数据长度）

        FNFirmwareUpgradeStatus fnUpgradeStatus;//固件升级实时状态回调函数
        void* pUpgradeStatusUserData;//fnUpgradeStatus的pUserData参数
}FirmwareUpgradeData;
```

描述：

固件包数据。

类型：

pData	指向固件数据（完整的固件数据首地址）
nDataLen	pData 内固件数据长度（完整的固件数据长度）
fnUpgradeStatus	固件升级实时状态回调函数
pUpgradeStatusUserData	fnUpgradeStatus 的 pUserData 参数

4.22 RebootDev

原型：

```
typedef struct tagRebootDev
{
        UINT8 byRes[4];//字节对齐，预留
}RebootDev;
```

描述：

设备重启。

类型：

byRes	字节对齐，预留

4.23 MasterSlaveSyncTime

原型：

```
typedef struct tagMasterSlaveSyncTime
{
        UINT64 hostSendTimestamp;//主机发送命令的时间（主机的本地时间）
        UINT64 slaveRecvTimestamp;//从机接收到命令的时间（从机的本地的时间）
        UINT64 slaveSendTimestamp;//从机发送命令的时间（从机的本地时间）
        UINT64 hostRecvTimestamp;//主机接收到命令的时间（主机的本地的时间）

}MasterSlaveSyncTime;
```

描述：

双目相机参数。

类型：

hostSendTimestamp	主机发送命令的时间（主机的本地时间）
slaveRecvTimestamp	从机接收到命令的时间（从机的本地的时间）
slaveSendTimestamp	从机发送命令的时间（从机的本地时间）
hostRecvTimestamp	主机接收到命令的时间（主机的本地的时间）

4.24 TofAnalogGain

原型：

```
typedef struct tagTofAnalogGain
{
        SBOOL  bAuto;//是否自动
        UINT8  szRes[2];//4 字节对齐，预留
```

```
    SBOOL  bUpdataValue;//是否更新增益值到板子（该字段仅在设置时有效）
    SINT32 lCurrent;//当前值

    SINT32 lDefault;//默认值（该字段仅在获取时有效）
    SINT32 lStep;//步进值（该字段仅在获取时有效）
    SINT32 lMax;//最大值（该字段仅在获取时有效）
    SINT32 lMin;//最小值（该字段仅在获取时有效）
}TofAnalogGain;
```

描述：

TOF 模拟增益。

类型：

bAuto	是否自动
szRes	4 字节对齐，预留
bUpdataValue	是否更新增益值到板子（该字段仅在设置时有效）
lCurrent	当前值
lDefault	默认值（该字段仅在获取时有效）
lStep	步进值（该字段仅在获取时有效）
lMax	最大值（该字段仅在获取时有效）
lMin	最小值（该字段仅在获取时有效）

4.25 TofDigitalGain

原型：

```
typedef struct tagTofDigitalGain
{
    SBOOL  bAuto;//是否自动
    UINT8  szRes[2];//4 字节对齐，预留
    SBOOL  bUpdataValue;//是否更新增益值到板子（该字段仅在设置时有效）
    SINT32 lCurrent;//当前值

    SINT32 lDefault;//默认值（该字段仅在获取时有效）
    SINT32 lStep;//步进值（该字段仅在获取时有效）
    SINT32 lMax;//最大值（该字段仅在获取时有效）
    SINT32 lMin;//最小值（该字段仅在获取时有效）
}TofDigitalGain;
```

描述：

TOF 数字增益。

类型：

bAuto	是否自动

szRes	4 字节对齐，预留
bUpdataValue	是否更新增益值到板子（该字段仅在设置时有效）
lCurrent	当前值
lDefault	默认值（该字段仅在获取时有效）
lStep	步进值（该字段仅在获取时有效）
lMax	最大值（该字段仅在获取时有效）
lMin	最小值（该字段仅在获取时有效）

4.26 TofFrameDataPixelOffset

原型：

```
typedef struct tagTofFrameDataPixelOffset
{
        UINT32 nOffset;//偏移量（像素个数）

}TofFrameDataPixelOffset;
```

描述：

TOF 回调函数里输出的 TOF 数据相对于 RAW 数据的像素偏移个数。

类型：

nOffset	偏移量（像素个数）

4.27 TofSensorStatus

原型：

```
//TOF Sensor状态
typedef enum tagTofSensorStatus
{
        TofSensorStatus_StreamOff = 1,//Sensor不出流
        TofSensorStatus_StreamOn  = 2,//Sensor出流

}TofSensorStatus;
```

描述：

TOF Sensor 状态。

类型：

TofSensorStatus_StreamOff	Sensor 不出流
TofSensorStatus_StreamOn	Sensor 出流

4.28 TofSensorStatusCtrl

原型：

```
typedef struct tagTofSensorStatusCtrl
{
        TofSensorStatus status;

}TofSensorStatusCtrl;
```

描述：

TOF Sensor 状态控制参数。

类型：

status	TOF Sensor 状态

4.29 RgbSensorStatusCtrl

原型：

```
typedef struct tagRgbSensorStatusCtrl
{
        UINT8 szRes[4];//预留

}RgbSensorStatusCtrl;
```

描述：

RGB Sensor 状态控制参数。

类型：

szRes	预留

4.30 SensorStatusCtrl

原型：

```
typedef struct tagSensorStatusCtrl
{
        UINT32 nIndex;//1---struTof有效, 2---struRgb有效

        union
        {
                TofSensorStatusCtrl struTof;//TOF Sensor状态控制参数

                RgbSensorStatusCtrl struRgb;//RGB Sensor状态控制参数
```

```
    }uParam;

}SensorStatusCtrl;
```

描述：

Sensor 状态控制。

类型：

nIndex	1---struTof 有效, 2---struRgb 有效
struTof	TOF Sensor 状态控制参数
struRgb	RGB Sensor 状态控制参数

4.31 TOF_DEV_PARAM_TYPE

原型：

```
typedef enum tagTOF_DEV_PARAM_TYPE
{
    TOF_DEV_PARAM_Temperature          = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x00),//温度信息
    TOF_DEV_PARAM_TofLensParameter     = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x01),//TOF模组内参和畸变（V1.0 版本，建议不要再用，因为不能适用于鱼眼模型）
    TOF_DEV_PARAM_TofCalibData         = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x02),//TOF模组标定数据
    TOF_DEV_PARAM_netdevinfo           = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x03),//网络接入设备信息
    TOF_DEV_PARAM_ReplaceTofCalibData  = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x04),//替换SDK里TOF模组标定数据（仅仅是替换SDK里标定数据，并非烧写到模组）
    TOF_DEV_PARAM_RemoteCapture        = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x05), //远程抓图：控制模块抓取数据并保存在模块内部；
    TOF_DEV_PARAM_ExportRaw            = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x06), //导出一帧RAW数据：实时的从模块里导出一帧RAW数据（适用于RAW数据和深度数据异步传输的情况）；
    TOF_DEV_PARAM_RgbLensParameter     = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x07),//RGB模组内参和畸变
    TOF_DEV_PARAM_UpgradeFirmware      = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x08),//升级固件
    TOF_DEV_PARAM_RebootDev            = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x09),//设备重启
    TOF_DEV_PARAM_StereoLensParameter  = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x0a),//双目相机参数
    TOF_DEV_PARAM_GetMasterSlaveSyncTime = MAKE_UNIQUE_ID(0x00, 0x00,
0x00, 0x0b),//获取主从机间同步时间
    TOF_DEV_PARAM_TofAnalogGain        = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x0c),//TOF模拟增益
    TOF_DEV_PARAM_TofDigitalGain       = MAKE_UNIQUE_ID(0x00, 0x00, 0x00,
0x0d),//TOF数字增益
    TOF_DEV_PARAM_TofLensParameterV20  = MAKE_UNIQUE_ID(0x00, 0x00,
```

0x00, 0x0e),//TOF模组内参和畸变（V2.0 版本）
TOF_DEV_PARAM_TofFrameDataPixelOffset= MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x0f),//TOF回调函数里输出的TOF数据相对于RAW数据的像素偏移个数
TOF_DEV_PARAM_DepthCalRoi = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x10),//深度计算的区域
TOF_DEV_PARAM_SensorStatusCtrl = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x11),//Sensor状态控制

}TOF_DEV_PARAM_TYPE;

描述：

设备参数类型。

类型：

TOF_DEV_PARAM_Temperature	温度信息
TOF_DEV_PARAM_TofLensParameter	TOF 模组内参和畸变（V1.0 版本，建议不要再用，因为不能适用于鱼眼模型）
TOF_DEV_PARAM_RgbLensParameter	RGB 模组内参和畸变
TOF_DEV_PARAM_TofCalibData	TOF 模组标定数据
TOF_DEV_PARAM_netdevinfo	网络接入设备信息
TOF_DEV_PARAM_ReplaceTofCalibData	替换 SDK 里 TOF 模组标定数据（仅仅是替换 SDK 里标定数据，并非烧写到模组）
TOF_DEV_PARAM_RemoteCapture	远程抓图：控制模块抓取数据并保存在模块内部；
TOF_DEV_PARAM_ExportRaw	导出一帧RAW数据：实时的从模块里导出一帧RAW数据（适用于RAW数据和深度数据异步传输的情况）
TOF_DEV_PARAM_UpgradeFirmware	升级固件
TOF_DEV_PARAM_RebootDev	设备重启
TOF_DEV_PARAM_StereoLensParameter	双目相机参数
TOF_DEV_PARAM_GetMasterSlaveSyncTime	获取主从机间同步时间
TOF_DEV_PARAM_TofAnalogGain	TOF模拟增益
TOF_DEV_PARAM_TofDigitalGain	TOF数字增益
TOF_DEV_PARAM_TofLensParameterV20	TOF模组内参和畸变（V2.0 版本）
TOF_DEV_PARAM_TofFrameDataPixelOffset	TOF回调函数里输出的TOF数据相对于RAW数据的像素偏移个数
TOF_DEV_PARAM_DepthCalRoi	深度计算的区域
TOF_DEV_PARAM_SensorStatusCtrl	Sensor状态控制

4.32 TofDeviceParamV20

原型：

```
typedef struct tagTofDeviceParamV20
{
        TOF_DEV_PARAM_TYPE type;//输入参数，只读

        union
        {
                TofDeviceTemperature   struTemperature;//温度信息【当type为TOF_DEV_PARAM_Temperature时有效】
                TofModuleLensParameter struTofLensParameter;//TOF模组内参和畸变【当type为TOF_DEV_PARAM_TofLensParameter时有效】（V1.0 版本，建议不要再用，因为不能适用于鱼眼模型）
                TofModuleLensParameterV20 struTofLensParameterV20;//TOF模组内参和畸变【当type为TOF_DEV_PARAM_TofLensParameterV20 时有效】（V2.0 版本）
                RgbModuleLensParameter struRgbLensParameter;//RGB模组内参和畸变【当type为TOF_DEV_PARAM_RgbLensParameter时有效】
                TofCalibData        struTofCalibData;//TOF模组标定数据【当type为TOF_DEV_PARAM_TofCalibData时有效】
                NetDevInfo_t        stuNetDevData;//网络接入设备信息【当type为TOF_DEV_PARAM_netdevinfo时有效】
                TofCalibData        struReplaceTofCalibData;//替换SDK里TOF模组标定数据【当type为TOF_DEV_PARAM_ReplaceTofCalibData时有效】
                RemoteCapture       struRemoteCapture;//远程抓图：控制模块抓取数据并保存在模块内部；【当type为TOF_DEV_PARAM_RemoteCapture时有效】
                TofRawData          struExportRaw;//导出一帧RAW数据：实时的从模块里导出一帧RAW数据（适用于RAW数据和深度数据异步传输的情况）；【当type为TOF_DEV_PARAM_ExportRaw时有效】
                FirmwareUpgradeData   struFirmware;//固件升级数据【当type为TOF_DEV_PARAM_UpgradeFirmware时有效】
                RebootDev           struRebootDev;//设备重启【当type为TOF_DEV_PARAM_RebootDev时有效】
                StereoLensParameter   struStereoLensParameter;//双目相机参数【当type为TOF_DEV_PARAM_StereoLensParameter时有效】
                MasterSlaveSyncTime   struMasterSlaveSyncTime;//主从机间同步时间【当type为TOF_DEV_PARAM_GetMasterSlaveSyncTime时有效】
                TofAnalogGain       struTofAnalogGain;//TOF模拟增益【当type为TOF_DEV_PARAM_TofAnalogGain时有效】
                TofDigitalGain      struTofDigitalGain;//TOF数字增益【当type为TOF_DEV_PARAM_TofDigitalGain时有效】
                TofFrameDataPixelOffset struPixelOffset;//TOF回调函数里输出的TOF数据相对于RAW数据的像素偏移个数【当type为TOF_DEV_PARAM_TofFrameDataPixelOffset时有效】
                DepthCalRoi         struDepthCalRoi;//深度计算的区域【当type为TOF_DEV_PARAM_DepthCalRoi时有效】
                SensorStatusCtrl      struSensorStatusCtrl;//Sensor状态控制【当type为TOF_DEV_PARAM_SensorStatusCtrl时有效】

        }uParam;
```

}TofDeviceParamV20;

描述：

设备参数（2.0 版本数据结构）。

类型：

type	指定的设备参数类型，输入参数，只读
struTemperature	温度信息【当 type 为 TOF_DEV_PARAM_Temperature 时有效】
struTofLensParameter	TOF 模组内参和畸变【当 type 为 TOF_DEV_PARAM_TofLensParameter 时有效】（V1.0 版本，建议不要再用，因为不能适用于鱼眼模型）
struTofLensParameterV20	TOF 模组内参和畸变【当 type 为 TOF_DEV_PARAM_TofLensParameterV20 时有效】（V2.0 版本）
struRgbLensParameter	RGB 模组内参和畸变【当 type 为 TOF_DEV_PARAM_RgbLensParameter 时有效】
struTofCalibData	TOF 模组标定数据【当 type 为 TOF_DEV_PARAM_TofCalibData 时有效】
stuNetDevData	网络接入设备信息【当 type 为 TOF_DEV_PARAM_netdevinfo 时有效】
struReplaceTofCalibData	替换 SDK 里 TOF 模组标定数据【当 type 为 TOF_DEV_PARAM_ReplaceTofCalibData 时有效】
struRemoteCapture	远程抓图：控制模块抓取数据并保存在模块内部；【当 type 为 TOF_DEV_PARAM_RemoteCapture 时有效】
struExportRaw	导出一帧 RAW 数据：实时的从模块里导出一帧 RAW 数据（适用于 RAW 数据和深度数据异步传输的情况）；【当 type 为 TOF_DEV_PARAM_ExportRaw 时有效】
struFirmware	固件升级数据【当 type 为 TOF_DEV_PARAM_UpgradeFirmware 时有效】
struRebootDev	设备重启【当 type 为 TOF_DEV_PARAM_RebootDev 时有效】
struStereoLensParameter	双目相机参数【当 type 为 TOF_DEV_PARAM_StereoLensParameter 时有效】
struMasterSlaveSyncTime	主从机间同步时间【当 type 为 TOF_DEV_PARAM_GetMasterSlaveSyncTime 时有效】
struTofAnalogGain	TOF 模拟增益【当 type 为 TOF_DEV_PARAM_TofAnalogGain 时有效】
struTofDigitalGain	TOF 数字增益【当 type 为 TOF_DEV_PARAM_TofDigitalGain 时有效】

struPixelOffset	TOF 回调函数里输出的 TOF 数据相对于 RAW 数据的像素偏移个数【当 type 为 TOF_DEV_PARAM_TofFrameDataPixelOffset 时有效】
struDepthCalRoi	深度计算的区域【当 type 为 TOF_DEV_PARAM_DepthCalRoi 时有效】
struSensorStatusCtrl	Sensor 状态控制【当 type 为 TOF_DEV_PARAM_SensorStatusCtrl 时有效】

4.33 TOFDEV_STATUS

原型：

```
typedef enum tagTOFDEV_STATUS
{
        TOFDEV_STATUS_UNUSED            = MAKE_UNIQUE_ID('U', 'U', 'S', 'E'),//（该值未使用，有效的设备状态从 1 开始）

        TOFDEV_STATUS_DEV_BROKEN        = MAKE_UNIQUE_ID('D', 'E', 'V', 'B'),//设备异常断开
        //
        TOFDEV_STATUS_READ_CALIB_DATA_SUC   = MAKE_UNIQUE_ID('R', 'C', 'D', 'S'),//读取标定数据成功
        TOFDEV_STATUS_READ_CALIB_DATA_FAILED = MAKE_UNIQUE_ID('R', 'C', 'D', 'F'),//读取标定数据失败
        //
        TOFDEV_STATUS_TOF_STREAM_FAILED     = MAKE_UNIQUE_ID('T', 'S', 'F', 0x00),//取TOF流失败

}TOFDEV_STATUS;
```

描述：

TOF 设备状态，设备可能处于 TOF、RGB、RGBD、IMU 数据流打开状态的并集。

参数：

TOFDEV_STATUS_UNUSED	该值未使用，有效的设备状态从 1 开始
TOFDEV_STATUS_DEV_BROKEN	设备异常断开
TOFDEV_STATUS_READ_CALIB_DATA_SUC	读取标定数据成功
TOFDEV_STATUS_READ_CALIB_DATA_FAILED	读取标定数据失败
TOFDEV_STATUS_TOF_STREAM_FAILED	取 TOF 流失败

4.34 HTOFD

原型：

```
typedef void* HTOFD;
```

描述：

TOF 设备的句柄，此句柄指向 TOF SDK 内部设备管理的内存区。

4.35 FNTofStream

原型：

typedef void (*FNTofStream)(TofFrameData *tofFrameData, void* pUserData);

描述：

TOF 输出点云与 IR 图像数据的回调函数。

参数：

tofFrameData	TOF 点云与 IR 图像数据结构指针，参加本章 TofFrameData
pUserData	用户数据指针，与 TOFD_StartTofStream 的 pUserData 是同一个。

4.36 FNTofDeviceStatus

原型：

typedef void (*FNTofDeviceStatus)(TOFDEV_STATUS tofDevStatus, void* pUserData);

描述：

TOF 设备状态的回调函数。

参数：

tofDevStatus	TOF 设备状态，参加本章 TOFDEV_STATUS
pUserData	用户数据指针。

4.37 FNRgbStream

原型：

typedef void (*FNRgbStream)(RgbFrameData *rgbFrameData, void* pUserData);

描述：

TOF 输出 RGB 图像数据的回调函数。

参数：

rgbFrameData	RGB 数据结构指针，参加本章 RgbFrameData
pUserData	用户数据指针，与 TOFD_StartRgbStream 的 pUserData 是同一个。

4.38 FNImuStream

原型：

typedef void (*FNImuStream)(ImuFrameData *imuFrameData, void* pUserData);

描述：

TOF 输出 IMU 数据的回调函数。

参数：

imuFrameData	IMU 数据结构指针，参加本章 ImuFrameData
pUserData	用户数据指针，与 TOFD_StartImuStream 的 pUserData 是同一个。

5 示例代码

详见 SDK 包里的 demo 程序源文件