Device SDK

User Manual

V4.4.41

History

Version	Date	Modifier	Summary of changes
Older versions	…	xrjiang /zhangyan	See the manual in the older version SDK.
4.4.30	11/18/2021	xrjiang	Add new word bWeakAuthority to TofDevInitParam; Add TOFD_OpenDevice_WithFd. Add TOF_DEV_SEEKER07C, TOF_DEV_SEEKER08A.
4.4.31	12/14/2021	xrjiang	Add TOF_DEV_DEMO_UPG.
4.4.32	01/05/2022	xrjiang	Add new word pDepthData, pDepthDataFilter to TofFrameData. Add new data type: TofFrameDataPixelOffset.
4.4.33	01/10/2022	xrjiang	Add new word bDisablePixelOffset to TofDevInitParam. Take the data structure DepthCalRoi as the public data type.
4.4.34	01/19/2022	xrjiang	Add TOF_DEV_CLEANER01G1, TOF_DEV_DEMO_C00P01A_NET, TOF_DEV_LOGITECH_C525,TOF_DEV_CHROMEBOOK.
4.4.35	02/14/2022	xrjiang	Add TOF_DEV_CLEANER01X.
4.4.36	02/21/2022	xrjiang	Add new parameter API: TOF_DEV_PARAM_SensorStatusCtrl.
4.4.37	03/14/2022	xrjiang	Add new TOF Filter: TOF_FILTER_RadialFusion.
4.4.38	03/25/2022	xrjiang	Add TOF_DEV_CLEANER01F1
4.4.39	04/01/2022	xrjiang	Take some data structure as the public data type. Add TOF_DEV_MARS01H
4.4.40	04/18/2022	xrjiang	Specifying custom log files is supported.
4.4.41	05/16/2022	xrjiang	Add some value for TOF_MODE. Update the definition of RgbDData

Catalog

1 Summary

1.1 Introduce

Sunny TOF device is a 3D camera product of machine vision. Its main functions are as fallow:

1. Real-time 3D position information measurement.
2. Real-time output of IR image data.
3. Support AE exposure and manual exposure modes (some TOF devices and TOF modules support this function).
4. Support HDRZ function (some TOF devices and TOF modules support this function).
5. Support multiple filtering functions (different TOF devices and TOF modules support different filtering functions).
6. Support multiple modes (different TOF devices and TOF modules support different modes).
7. Support real-time RGB image acquisition (some TOF modules have this function).
8. Support real-time RGBD data acquisition (some TOF modules support this function).
9. Support IMU data real-time acquisition (some TOF modules support this function).
10. Support Windows7, Linux(Ubuntu…), Android…

The TOF module products supported by this SDK include:

1、EPC

2、MARS01A

3、MARS01B

4、MARS04A

5、MARS04B

6、MARS05

7、MARS05A

8、Other products not listed one by one.

1.2 Contents contained in this SDK

- Librarys files, header files, parameter files for special platforms;
- API user manual document [this document];
- SDK sample code;
- Building script based on CMake;

1.3 Supported platforms

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

2 Interface description of TOF device SDK

2.1 TOFD_Init

Function prototype:

TOFDDLL TOFRET   TOFD_Init(TofDevInitParam* pInitParam);

Function description:
Initialize the TOF device. Other functions of the TOF SDK can only be called after this function call.

Function parameters:

pInitParam	[input]	It is the initialization parameter of SDK library which cannot be NULL.

Return value:
If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.2 TOFD_Uninit

Function prototype:

TOFDDLL TOFRET   TOFD_Uninit(void);

Function description:
Stop the output of all data (including TOF point cloud data, IR image data, RGB data and IMU data) and clear the relevant memory resources. This function is usually called when the main function of the application exits. After this function is called, other functions of the TOF module SDK cannot be called.

2.3 TOFD_GetSDKVersion

Function prototype:

TOFDDLL SCHAR*   TOFD_GetSDKVersion(void);

Function description:
Get the SDK version information (the return value is a string version number).

Return value:
The SDK version information is returned after the function is executed successfully.

2.4 TOFD_SearchDevice

Function prototype:

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

Function description:
Search all TOF devices connected to this system.

Function parameters:

ppDevsDesc	[output]	Please refer to the structure TofDeviceDescriptor in Chapter 4. It cannot be NULL.

pDevNum	[output]	It is the number of devices which cannot be NULL.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.5 TOFD_OpenDevice

Function prototype:

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

Function description:

This function is used to open a TOF module device. The operation functions of the device, including TOF point cloud and IR image acquisition, RGB image acquisition, rgbd image acquisition, IMU data acquisition, TOF exposure and filter setting, must be executed after this function is called.

Function parameter:

pDevDesc	[input]	Please refer to the structure TofDeviceDescriptor in Chapter 4. This parameter can only be obtained through functions TOFD_SearchDevice and cannot be NULL.
fnTofDevStatus	[input]	Please refer to the callback function FNTofDeviceStatus in Chapter 4,which is the status callback function of TOF module device.When the device status changes, the application will be notified through this function. This parameter can be NULL.
pUserData	[input]	It is the the pointer of the user data, which will be passed to the upper application as a parameter of fnTofDevStatus.

Return value:

The function returns the handle of the device after successful execution, otherwise it returns NULL.

2.6 TOFD_OpenDevice_WithFd

Function prototype:

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

Function description:

This function is used to open a TOF module device. The operation functions of the device, including TOF point cloud and IR image acquisition, RGB image acquisition, rgbd image acquisition, IMU data acquisition, TOF exposure and filter setting, must be executed after this function is called.

Function parameter:

pDevDesc	[input]	Please refer to the structure TofDeviceDescriptor in Chapter 4. This parameter can only be obtained through functions TOFD_SearchDevice and cannot be NULL.
fnTofDevStatus	[input]	Please refer to the callback function FNTofDeviceStatus in Chapter 4,which is the status callback function of TOF module device.When the device status changes, the application will be notified through this function. This parameter can be NULL.

pUserData	[input]	It is the the pointer of the user data, which will be passed to the upper application as a parameter of fnTofDevStatus.

Return value:

The function returns the handle of the device after successful execution, otherwise it returns NULL.

2.7 TOFD_CloseDevice

Function prototype:

TOFDDLL TOFRET   TOFD_CloseDevice(HTOFD hTofDev);

Function description:

Close the TOF device. The operation functions of the device, including TOF point cloud and IR image acquisition, RGB image acquisition, RGBD image acquisition, IMU data acquisition, TOF exposure, and filter setting, must be executed before the function is called.

Function parameter

hTofDev	[input]	It is the TOF device handle, cannot be NULL.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.8 TOFD_GetDeviceInfo

Function prototype:

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

Function description:

Obtain equipment information (generally indicating equipment capability).

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
pTofDeviceInfo	[output]	It is the obtained device information,which can be read only.。

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.9 TOFD_GetDeviceParam

Function prototype:

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

Function description:

Get equipment parameters (equipment support is required) (it has been gradually abandoned).

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
pTofDeviceParam	[output]	It is the obtained device information.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.10 TOFD_SetDeviceParam

Function prototype:

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

Function description:

Set equipment parameters (equipment support is required) (it has been gradually abandoned).

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
pTofDeviceParam	[input]	It is the parameter that needs to be set to the device.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.11 TOFD_GetDeviceParamV20

Function prototype:

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

Function description:

Gets the device parameter of the specified type (version 2.0 interface).

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
pTofDeviceParam	[input/_output]	It is the obtained device parameter. In pTofDeviceParam, type is the input parameter and uParam is the output parameter. 1) When type is TOF_DEV_PARAM_Temperature, struTemperature in uParam is valid. 2) When type is TOF_DEV_PARAM_TofLensParameter, struTofLensParameter in uParam is valid. 3) When type is TOF_DEV_PARAM_TofCalibData, struTofCalibData in uParam is valid. 4) When type is TOF_DEV_PARAM_netdevinfo, stuNetDevData in uParam is valid.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.12 TOFD_SetDeviceParamV20

Function prototype:

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

Function description:

Set the device parameters of the specified type (version 2.0 interface).

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
pTofDeviceParam	[input/ output]	It is the obtained device parameter. In pTofDeviceParam, type is the input parameter and uParam is the output parameter. 1) When type is TOF_DEV_PARAM_Temperature, struTemperature in uParam is valid. 2) When type is TOF_DEV_PARAM_TofLensParameter, struTofLensParameter in uParam is valid. 3) When type is TOF_DEV_PARAM_TofCalibData, struTofCalibData in uParam is valid. 4) When type is TOF_DEV_PARAM_netdevinfo, stuNetDevData in uParam is valid.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.13 TOFD_SetTofAE

Function prototype:

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

Function description:

Set TOF exposure mode (manual mode or automatic mode).

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
bEnable	[input]	It is a parameter to judge whether it is an automatic exposure mode. When it is true, it is auto exposure mode. When it is false, it is in manual exposure mode.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.14 TOFD_SetTofExpTime

Function prototype:

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

Function description:
Sets the current exposure time of TOF.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
expTime	[input]	It represents exposure time of TOF. This parameter must be within the range of effective exposure time of TOF, which is obtained through the function TOFD_GetTofExpTime.

Return value:
If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.15 TOFD_GetTofExpTime

Function prototype:

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

Function description:
Obtain the exposure time parameter of TOF.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
pExp	[output]	It is the TOF exposure time parameter, which cannot be NULL.

Return value:
If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.16 TOFD_SetTofFilter

Function prototype:

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

Function description:
Turns the TOF filter of the specified type on or off.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
type	[input]	It represents the filter type.
bEnable	[input]	It is a parameter representing whether TOF filtering is turned on. True is on and false is off.

Return value:
If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.17 TOFD_GetTofFilter

Function prototype:

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

Function description:
Get the switch status of the TOF filter of the specified type.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
type	[input]	It represents the filter type.
pbEnable	[output]	It is a parameter representing whether TOF filtering is turned on. True is on and false is off. It cannot be NULL.

Return value:
If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.18 TOFD_SetTofHDRZ

Function prototype:

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

Function description:
Turn on or turn off TOF HDRZ.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
bEnable	[input]	It is a parameter representing whether TOF HDRZ is turned on. True is on and false is off.

Return value:
If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.19 TOFD_SetTofRemoveINS

Function prototype:

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

Function description:
Turn on or turn off the algorithm TOF RemoveINS.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
bEnable	[input]	It is a parameter representing whether TOF RemoveINS is turned on. True is on and false is off.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.20 TOFD_SetTofMPIFlag

Function prototype:

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

Function description:

Turn on or turn off the algorithm TOF MPIFlag (This function has been obsoleted. Please use TOFD_SetTofFilter(xxx, TOF_FILTER_MPIFilter, xxx).).

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
bEnable	[input]	It is a parameter representing whether TOF MPIFlag is turned on. True is on and false is off.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.21 TOFD_StartTofStream

Function prototype:

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

Function description:

Start real-time acquisition of TOF point cloud data and IR image data.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
tofMode	[input]	It represents the TOF mode, which is obtained through the function TOFD_SearchDevice. Please refer to the description of enumerating cases of TOF_MODE in Chapter 4.
fnTofStream	[input]	It is a callback function for outputting TOF point cloud data and IR image data. This parameter cannot be NULL.
pUserData	[input]	It is the user data pointer, and this parameter will be output to the application as a parameter of fnTofStream.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS or TOFRET_SUCCESS_READING_CALIB, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

SPECIAL NOTE: When the return value is TOFRET_SUCCESS_READING_CALIB, the status will generally be recalled back through FNTofDeviceStatus before the data flow is called back.

2.22 TOFD_StopTofStream

Function prototype:

TOFDDLL TOFRET   TOFD_StopTofStream(HTOFD hTofDev);

Function description:
Stop acquiring TOF point cloud data and IR image data in real time.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.

Return value:
If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.23 TOFD_GetRgbProperty

Function prototype:

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

Function description:
Gets the parameter value of the specified attribute of the RGB module.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
Property	[input]	It represents the specified RGB attribute.
pValue	[output]	It represents the parameter value of the specified RGB attribute obtained, which cannot be NULL.

Return value:
If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.24 TOFD_SetRgbProperty

Function prototype:

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

Function description:
Set the parameter value of the specified attribute of the RGB module.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
Property	[input]	It represents the specified RGB attribute.
lValue	[input]	It represents the parameter value of the specified RGB attribute.
lFlag	[input]	It represents the attachment attribute of the parameter value of the specified RGB attribute set to indicate whether it is automatic or manual.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.25 TOFD_StartRgbStream

Function prototype:

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

Function description:

Start real-time acquisition of RGB image data.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
fnRgbStream	[input]	It represents a callback function that outputs RGB data. This parameter cannot be NULLl.
pUserData	[input]	It is a pointer to user data. This parameter will be output to the application as a parameter of fnRgbStream.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS or TOFRET_SUCCESS_READING_CALIB, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

SPECIAL NOTE: When the return value is TOFRET_SUCCESS_READING_CALIB, the status will generally be recalled back through FNTofDeviceStatus before the data flow is called back.

2.26 TOFD_StopRgbStream

Function prototype:

TOFDDLL TOFRET TOFD_StopRgbStream(HTOFD hTofDev);

Function description:

Stop acquiring RGB image data in real time.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.27 TOFD_StartImuStream

Function prototype:

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

Function description:

Start real-time acquisition of IMU data.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.
fnImuStream	[input]	It represents the callback function that outputs IMU data. This parameter cannot be NULL.
pUserData	[input]	It represents the user data pointer, which will be output to the application as a parameter of fnImuStream.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

2.28 TOFD_StopImuStream

Function prototype:

TOFDDLL TOFRET   TOFD_StopImuStream(HTOFD hTofDev);

Function description:

Stop real-time acquisition of IMU data.

Function parameter:

hTofDev	[input]	It is the TOF device handle, cannot be NULL.

Return value:

If the function is executed successfully, it returns TOFRET_SUCCESS, otherwise it returns other error values. Please refer to the description of TOFRET for specific error values.

3 Common Data structure and type definition

3.1 TOFRET

Prototype:

```
typedef enum tagTOFRET
{
	/** Success (no error) */
	TOFRET_SUCCESS = 0x00000000,
	/** Success (no error, and start to read calibration 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;
```

Description:

Definition of the error value of TOF SDK.

Parameter:

TOFRET_SUCCESS	Success
TOFRET_SUCCESS_READING_CALIB	Success, and start to read the calibration file
TOFRET_ERROR_IO	IO error
TOFRET_ERROR_INVALID_PARAM	Invalid parameter
TOFRET_ERROR_ACCESS	Device operation failed
TOFRET_ERROR_NO_DEVICE	Device does not exist
TOFRET_ERROR_TIMEOUT	Access to the device timeout
TOFRET_ERROR_OVERFLOW	Data overflow
TOFRET_ERROR_NO_MEM	Failed to allocate memory
TOFRET_ERROR_WRONG_STATUS	Status error
TOFRET_ERROR_NOT_SUPPORTED	Unsupported function
TOFRET_ERROR_ALREADY_IN_USE	The device is already in use, indicating that the device has been turned on
TOFRET_ERROR_DATA	Wrong data
TOFRET_ERROR_CFG_FILE_NOT_FOUND	Failed to find configuration file
TOFRET_ERROR_READ_CALIB_FAILED	Failed to read calibration file

TOFRET_ERROR_USB_WRITE	USB write error
TOFRET_ERROR_USB_READ	USB read error
TOFRET_ERROR_USB_DISCONNECT	USB disconnect
TOFRET_HAL_FAILED	Failed to access the HAL layer
TOFRET_HAL_UNSUPPORT	The HAL layer does not support this function
TOFRET_HAL_HARDWARE_UNRESPONSIVE	The hardware does not respond to the HAL layer
TOFRET_HAL_TIMEOUT	HAL access hardware timeout
TOFRET_HAL_INTERFACE_BOARD_NOT_SUPPORT	HAL interface board not supported
TOFRET_HAL_CONFIG_READ_FAILED	HAL Layer Read Configuration Error
TOFRET_HAL_MODULE_LOAD_FAILED	Failed to open module
TOFRET_HAL_MODULE_SYSMBOL_CALL_FAILED	Failed to call of the symbol table of HAL layer
TOFRET_HAL_OBJ_INSTANCE_FAILE	HAL layer target initialization error
TOFRET_HAL_CAMERA_NOT_FOUND	Failed to found the TOF device of HAL layer
TOFRET_HAL_INTERFACE_BOARD_SETTING_FAILED	Failed to set the interface board of HAL layer
TOFRET_HAL_IIC_READ_FAILED	Failed to read the I2C of HAL layer
TOFRET_HAL_IIC_WRITE_FAILED	Failed to write the I2C of HAL layer
TOFRET_HAL_IO_SETTING_FAILED	Failed to set the HAL layer
TOFRET_ERROR_OTHER	Other errors

3.2 MAKE_UNIQUE_ID

Prototype:

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

Description:

32-bit ID number generated by specific rules.

3.3 TOF_MODE

Prototype:

typedef enum tagTOF_MODE

```
{
     //Dual frequency
    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,

    //Single frequency
    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：These modes represent raw data with HDRZ fusion
    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,

    //Diffferent frequency
    TOF_MODE_5FPS = 0x00040000,
    TOF_MODE_10FPS = 0x00080000,
    TOF_MODE_20FPS = 0x00100000,
    TOF_MODE_30FPS = 0x00200000,
    TOF_MODE_45FPS = 0x00400000,
    TOF_MODE_60FPS = 0x00800000,

    //Name to be determined
    TOF_MODE_ADI_1M5 = 0x01000000,
    TOF_MODE_ADI_5M = 0x02000000,

    //Custom
    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
    TOF_MODE_DEBUG          = 0x80000000,

}TOF_MODE;
```

Description:

TOF_MODE enumeration defines TOF mode.

Type:

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

Prototype:

```
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;
```

Description:

Types of TOF data filtering.

Type:

TOF_FILTER_RemoveFlyingPixel	Remove flying pixel filtering
TOF_FILTER_AdaptiveNoiseFilter	Adaptive noise filtering
TOF_FILTER_InterFrameFilter	Inter-frame filtering
TOF_FILTER_PointCloudFilter	Point cloud filtering
TOF_FILTER_StraylightFilter	Stray light filtering
TOF_FILTER_CalcIntensities	CalcIntensities filtering
TOF_FILTER_MPIFlagAverage	MPIFlagAverage filtering
TOF_FILTER_MPIFlagAmplitude	MPIFlagAmplitude filtering
TOF_FILTER_MPIFlagDistance	MPIFlagDistance filtering
TOF_FILTER_ValidateImage	ValidateImage filtering
TOF_FILTER_SparsePointCloud	Sparse point cloud filtering
TOF_FILTER_Average	Average filtering
TOF_FILTER_Median	Median filter
TOF_FILTER_Confidence	Confidence filtering
TOF_FILTER_MPIFilter	MPI filtering
TOF_FILTER_PointCloudCorrect	Point cloud correction filtering
TOF_FILTER_LineRecognition	Black line detection filtering

TOF_FILTER_RadialFusion	Radial Fusion filtering

3.5 TofFilterCfg_RemoveFlyingPixel

Prototype:

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

Description:

Parameters of Remove flying pixel filtering.

Type:

f0	Parameter f0
f1	Parameter f1
nd	Parameter nd
fd	Parameter fd

3.6 TofFilterCfg_AdaptiveNoiseFilter

Prototype:

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

Description:

Parameters of Adaptive noise filtering.

Type:

k	Parameter k
s	Parameter s
t	Parameter t

3.7 TofFilterCfg_InterFrameFilter

Prototype:

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

Description:

Parameters of Inter-frame filtering.

Type:

mdg	Parameter mdg
mdt	Parameter mdt
fg1	Parameter fg1
fg2	Parameter fg2

3.8 TofFilterCfg_PointCloudFilter

Prototype:

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

Description:

Parameters of Point cloud filtering.

Type:

k	Parameter k

3.9 TofFilterCfg_StraylightFilter

Prototype:

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

Description:

Parameters of Stray light filtering.

Type:

d	Parameter d
t	Parameter t

3.10 TofFilterCfg_CalcIntensities

Prototype:

```
typedef struct tagTofFilterCfg_CalcIntensities
{
	UINT8 szRes[4];//Reserve 4 bytes for alignment
}TofFilterCfg_CalcIntensities;
```

Description:

Parameter of CalcIntensities filtering.

Type:

szRes	Reserve

3.11 TofFilterCfg_MPIFlagAverage

Prototype:

```
typedef struct tagTofFilterCfg_MPIFlagAverage
{
	UINT8 szRes[4];//Reserve 4 bytes for alignment
}TofFilterCfg_MPIFlagAverage;
```

Description:

Parameters of MPIFlagAverage filtering.

Type:

szRes	Reserve

3.12 TofFilterCfg_MPIFlagAmplitude

Prototype:

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

Description:

Parameters of MPIFlagAmplitude filtering.

Type:

mat	Parameter mat
ndt	Parameter ndt

3.13 TofFilterCfg_MPIFlagDistance

Prototype:

```
typedef struct tagTofFilterCfg_MPIFlagDistance
{
	UINT8 szRes[4];//Reserve 4 bytes for alignment
}TofFilterCfg_MPIFlagDistance;
```

Description:

Parameters of MPIFlagDistance filtering.

Type:

szRes	Reserve

3.14 TofFilterCfg_ValidateImage

Prototype:

```
typedef struct tagTofFilterCfg_ValidateImage
{
	UINT8 szRes[4];//Reserve 4 bytes for alignment
}TofFilterCfg_ValidateImage;
```

Description:

Parameters of ValidateImage filtering.

Type:

szRes	Reserve

3.15 TofFilterCfg_SparsePointCloud

Prototype:

```
typedef struct tagTofFilterCfg_SparsePointCloud
{
	UINT8 szRes[4];//Reserve 4 bytes for alignment
}TofFilterCfg_SparsePointCloud;
```

Description:

Parameters of Sparse point cloud filtering.

Type:

szRes	Reserve

3.16 TofFilterCfg_Average

Prototype:

```
typedef struct tagTofFilterCfg_Average
{
	UINT8 szRes[4];//Reserve 4 bytes for alignment
}TofFilterCfg_Average;
```

Description:

Parameters of Mean filtering.

Type:

szRes	Reserve

3.17 TofFilterCfg_Median

Prototype:

```
typedef struct tagTofFilterCfg_Median
{
	UINT8 szRes[4];//Reserve 4 bytes for alignment
}TofFilterCfg_Median;
```

Description:

Parameters of Median filter.

Type:

szRes	Reserve

3.18 TofFilterCfg_Confidence

Prototype:

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

Description:

Parameters of Confidence filtering.

Type:

t	Parameter t

3.19 TofFilterCfg_MPIFilter

Prototype:

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

Description:

Parameters of MPI filtering.

Type:

ndt	Parameter ndt
fdt	Parameter fdt
nnr	Parameter nnr
mnr	Parameter mnr
fnr	Parameter fnr
rd	Parameter rd

3.20 TofFilterCfg_PointCloudCorrect

Prototype:

```
typedef struct tagTofFilterCfg_PointCloudCorrect
```

```
{
	FLOAT32 da;//The angle of the module downslope
	FLOAT32 tgd;//The height of the module from the ground
	FLOAT32 t1;
	FLOAT32 t2;
}TofFilterCfg_PointCloudCorrect;
```

Description:

Parameters of Point cloud correction filtering.

Type:

da	Parameter da
tgd	Parameter tgd
t1	Parameter t1
T2	Parameter T2

3.21 TofFilterCfg_LineRecognition

Prototype:

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

Description:

Parameters of Black line detection filtering.

Type:

ht	Parameter ht
cgt	Parameter cgt
fgst	Parameter fgst
gstr	Parameter gstr
spgt	Parameter spgt
opgt	Parameter opgt
rc	Parameter rc
lc	Parameter lc
ma	Parameter ma

3.22 TofFilterCfg_RadialFusion

Prototype:

```
typedef struct tagTofFilterCfg_RadialFusion
{
	UINT8 szRes[4];//Reserve 4 bytes for alignment
}TofFilterCfg_RadialFusion;
```

Description:

Parameters of Black line detection filtering.

Type:

szRes	Reserve

3.23 TofFilterCfg

Prototype:

```
typedef struct tagTofFilterCfg
{
	TOF_FILTER type;//A certain type of filtering, and generally only input parameters (read-only)

	union
	{
		SBOOL bEnable;//Whether to enable, it can be input or output parameters (not open yet, currently an invalid field)
		UINT8 szRes[4];//Reserve 4 bytes for alignment
	}uRes;

	union
	{
		TofFilterCfg_RemoveFlyingPixel struRemoveFlyingPixel;//Valid when the type value is TOF_FILTER_RemoveFlyingPixel
		TofFilterCfg_AdaptiveNoiseFilter struAdaptiveNoiseFilter;//Valid when the type value is TOF_FILTER_AdaptiveNoiseFilter
		TofFilterCfg_InterFrameFilter struInterFrameFilter;//Valid when the type value is TOF_FILTER_InterFrameFilter
		TofFilterCfg_PointCloudFilter struPointCloudFilter;//Valid when the type value is TOF_FILTER_PointCloudFilter
		TofFilterCfg_StraylightFilter struStraylightFilter;//Valid when the type value is TOF_FILTER_StraylightFilter
		TofFilterCfg_CalcIntensities struCalcIntensities;//Valid when the type value is TOF_FILTER_CalcIntensities
		TofFilterCfg_MPIFlagAverage struMPIFlagAverage;//Valid when the type value is TOF_FILTER_MPIFlagAverage
		TofFilterCfg_MPIFlagAmplitude struMPIFlagAmplitude;//Valid when the type value is TOF_FILTER_MPIFlagAmplitude
		TofFilterCfg_MPIFlagDistance struMPIFlagDistance;//Valid when the type value is TOF_FILTER_MPIFlagDistance
		TofFilterCfg_ValidateImage struValidateImage;//Valid when the type value is
TOF_FILTER_ValidateImage
            TofFilterCfg_SparsePointCloud struSparsePointCloud;//Valid when the type value is
TOF_FILTER_SparsePointCloud
            TofFilterCfg_Average struAverage;//Valid when the type value is
TOF_FILTER_Average
            TofFilterCfg_Median struMedian;//Valid when the type value is TOF_FILTER_Median
            TofFilterCfg_Confidence struConfidence;//Valid when the type value is
TOF_FILTER_Confidence
            TofFilterCfg_MPIFilter struMPIFilter;//Valid when the type value is
TOF_FILTER_MPIFilter
            TofFilterCfg_PointCloudCorrect struPointCloudCorrect;//Valid when the type value is
TOF_FILTER_PointCloudCorrect
            TofFilterCfg_LineRecognition struLineRecognition;//Valid when the type value is
TOF_FILTER_LineRecognition
            TofFilterCfg_RadialFusion struRadialFusion;//Valid when the type value is
TOF_FILTER_RadialFusion
      }uCfg;//Specific configuration of a filtering type, either input or output parameters

}TofFilterCfg;
```

Description:

Detailed configuration of filtering parameters.

Type:

type		A certain type of filtering, and generally only input parameters (read-only)
uRes	bEnable	Whether to enable, it can be input or output parameters (not open yet, currently an invalid field)
	szRes	Reserve 4 bytes for alignment
uCfg		The filtering type is specified by type and the specific parameters are configured, according to the type, different fields are used
	struRemoveFlyingPixel	Valid when the type value is TOF_FILTER_RemoveFlyingPixel
	struAdaptiveNoiseFilter	Valid when the type value is TOF_FILTER_AdaptiveNoiseFilter
	struInterFrameFilter	Valid when the type value is TOF_FILTER_InterFrameFilter
	struPointCloudFilter	Valid when the type value is TOF_FILTER_PointCloudFilter
	struStraylightFilter	Valid when the type value is TOF_FILTER_StraylightFilter
	struCalcIntensities	Valid when the type value is TOF_FILTER_CalcIntensities
	struMPIFlagAverage	Valid when the type value is TOF_FILTER_MPIFlagAverage
	struMPIFlagAmplitude	Valid when the type value is TOF_FILTER_MPIFlagAmplitude
	struMPIFlagDistance	Valid when the type value is TOF_FILTER_MPIFlagDistance

	struValidateImage	Valid when the type value is TOF_FILTER_ValidateImage
	struSparsePointCloud	Valid when the type value is TOF_FILTER_SparsePointCloud
	struAverage	Valid when the type value is TOF_FILTER_Average
	struMedian	Valid when the type value is TOF_FILTER_Median
	struConfidence	Valid when the type value is TOF_FILTER_Confidence
	struMPIFilter	Valid when the type value is TOF_FILTER_MPIFilter
	struPointCloudCorrect	Valid when the type value is TOF_FILTER_PointCloudCorrect
	struLineRecognition	Valid when the type value is TOF_FILTER_LineRecognition
	struRadialFusion	Valid when the type value is TOF_FILTER_RadialFusion

3.24 EXP_MODE

Prototype:

```
typedef enum tagEXP_MODE
{
	EXP_MODE_MANUAL = 0x00000001,//Manual exposure
	EXP_MODE_AUTO = 0x00000002,//Automatic exposure (AE)
}EXP_MODE;
```

Description:

Types of TOF exposure.

Type:

EXP_MODE_MANUAL	Manual exposure
EXP_MODE_AUTO	Automatic exposure (AE)

3.25 GRAY_FORMAT

Prototype:

```
typedef enum tagGRAY_FORMAT
{
	GRAY_FORMAT_UINT8 = 0,//8-bit data
	GRAY_FORMAT_UINT16,//Unsigned 16-bit data
	GRAY_FORMAT_FLOAT,//Floating point data
	GRAY_FORMAT_BGRD,//32 bits per pixel, stored in the order of B, G, R, D

}GRAY_FORMAT;
```

Description:

Grayscale data format.

Type:

GRAY_FORMAT_UINT8	8-bit grayscale data
GRAY_FORMAT_UINT16	Unsigned 16-bit grayscale data
GRAY_FORMAT_FLOAT	Floating point grayscale data
GRAY_FORMAT_BGRD	32-bit grayscale data, 32 bits per pixel, stored in the order of B, G, R, D

3.26 PointData

Prototype:

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

Description:

Data structure of TOF point cloud.

Parameter:

x	X coordinate value of point cloud
y	Y coordinate value of point cloud
z	Z coordinate value of point cloud

3.27 RgbDData

Prototype:

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

Description:

Data Structure of TOF device RGBD.

Parameter:

b	Blue component of color
g	Green component of color
r	Red component of color

3.28 COLOR_FORMAT

Prototype:

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

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

	//YUV format
	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 format
	COLOR_FORMAT_BGR      = MAKE_UNIQUE_ID('B', 'G', 'R', 0x00), //RGB24（Each
pixel occupies 3 bytes, stored in the order of B, G, R）
	COLOR_FORMAT_RGB      = MAKE_UNIQUE_ID('R', 'G', 'B', 0x00), //RGB24（Each
pixel occupies 3 bytes, stored in the order of R, G, B）
	COLOR_FORMAT_BGRA     = MAKE_UNIQUE_ID('B', 'G', 'R', 'A'), //RGB32（Each
pixel occupies 4 bytes, stored in the order of B, G, R, A）
	COLOR_FORMAT_RGBA     = MAKE_UNIQUE_ID('R', 'G', 'B', 'A'), //RGB32（Each
pixel occupies 4 bytes, stored in the order of R, G, B, A）
}COLOR_FORMAT;
```

Description:

Types of RGB data format.

Type:

COLOR_FORMAT_MJPG	MJPG format
COLOR_FORMAT_H264	H264 format
COLOR_FORMAT_YUV422	YUV422 format
COLOR_FORMAT_YUYV	YUYV format
COLOR_FORMAT_I420	I420 format
COLOR_FORMAT_YV12	YV12 format
COLOR_FORMAT_NV12	NV12 format
COLOR_FORMAT_NV21	NV21 format
COLOR_FORMAT_BGR	RGB24（Each pixel occupies 3 bytes, stored in the order of B, G, R）
COLOR_FORMAT_RGB	RGB24（Each pixel occupies 3 bytes, stored in the order of R, G, B）
COLOR_FORMAT_BGRA	RGB32（Each pixel occupies 4 bytes, stored in the order of B, G, R, A）
COLOR_FORMAT_RGBA	RGB32（Each pixel occupies 4 bytes, stored in the order of R, G, B, A）

3.29 RgbData

Prototype:

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

Description:

Data Structure of TOF device RGB.

Parameter:

r	Red component of color
g	Green component of color
b	Blue component of color

3.30 RgbModuleLensParameter

Prototype:

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

Description:

Internal parameters and distortion parameters of the RGB module.

Type:

fx	Parameter fx
fy	Parameter fy
cx	Parameter cx
cy	Parameter cy
k1	Parameter k1
k2	Parameter k2
p1	Parameter p1

p2	Parameter p2
k3	Parameter k3

3.31 StereoLensParameter

Prototype:

```
typedef struct tagStereoLensParameter
{
	FLOAT32 szRotationMatrix[3][3];//Binocular rotation matrix
	FLOAT32 szTranslationMatrix[3];//Binocular translation matrix
}StereoLensParameter;
```

Description:

Parameters of binocular camera.

Type:

szRotationMatrix	Binocular rotation matrix
szTranslationMatrix	Binocular translation matrix

3.32 TofExpouse

Prototype:

```
typedef struct tagTofExpouse
{
	UINT32		nCurrent;//Current value, readable and writable
	UINT32		nDefault;//Default value, read only
	UINT32		nStep;//Step value, read only
	UINT32		nMax;//Maximum value, read only
	UINT32		nMin;//Minimum value, read only
}TofExpouse;
```

Description:

Parameters of TOF exposure.

Type:

nCurrent	Current value
nDefault	Default value
nStep	Step value
nMax	Maximum value
nMin	Minimum value

3.33 TofExpouseGroup1

Prototype:

```
typedef struct tagTofExpouseGroup1
{
	TofExpouse exp;//Exposure parameters
}TofExpouseGroup1;
```

Description:

Parameters of TOF exposure（Combination method 1）.

Type:

exp	Exposure parameters

3.34 TofExpouseGroup2

Prototype:

```
typedef struct tagTofExpouseGroup2
{
        TofExpouse exp_AEF;//Exposure parameters of automatic exposure frame
        TofExpouse exp_FEF;//Exposure parameters of fixed exposure frame
}TofExpouseGroup2;
```

Description:

Parameters of TOF exposure (Combination method 2).

Type:

exp_AEF	Exposure parameters of automatic exposure frame
exp_FEF	Exposure parameters of fixed exposure frame

3.35 TofExpouseGroup3

Prototype:

```
typedef struct tagTofExpouseGroup3
{
        TofExpouse exp_AEF;//Exposure parameters of automatic exposure frame
        TofExpouse exp_FEF;//Exposure parameters of fixed exposure frame
        TofExpouse exp_Gray;//Exposure parameters of gray exposure frame
}TofExpouseGroup3;
```

Description:

Parameters of TOF exposure (Combination method 3).

Type:

exp_AEF	Exposure parameters of automatic exposure frame
exp_FEF	Exposure parameters of fixed exposure frame
exp_Gray	Exposure parameters of gray exposure frame

3.36 TofExpouseItems

Prototype:

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

        union
        {
                //[Type 1]: Only applicable when raw data with single-frequency or dual-frequency
                TofExpouseGroup1 g1;//Exposure parameters

                //[Type 2]: Only applicable when raw data with automatic exposure frames and fixed
exposure frames（During intra-frame HDRZ fusion）
                TofExpouseGroup2 g2;//Exposure parameters

                //[Type 3]: Only applicable when raw data with automatic exposure frames and fixed
exposure frames（During intra-frame HDRZ fusion）, in addition, it is allowed to set gray exposure
frames
                TofExpouseGroup3 g3;// Exposure parameters
        }uParam;

}TofExpouseItems;
```

Description:

Combination of options of TOF exposure parameters.

Type:

nIndex	1---g1 valid, 2---g2 valid, 3---g3 valid
g1	Exposure parameters
g2	Exposure parameters
g3	Exposure parameters

3.37 TofExpouseCurrentGroup1

Prototype:

```
typedef struct tagTofExpouseCurrentGroup1
{
        UINT32 exp;//Exposure value
}TofExpouseCurrentGroup1;
```

Description:

TOF exposure value (Combination method 1).

Type:

exp	Exposure value

3.38 TofExpouseCurrentGroup2

Prototype:

```
typedef struct tagTofExpouseCurrentGroup2
{
        UINT32 exp_AEF;//Exposure value of automatic exposure frame
        UINT32 exp_FEF;//Exposure value of fixed exposure frame
}TofExpouseCurrentGroup2;
```

Description:

Value of TOF exposure (Combination method 2).

Type:

exp_AEF	Exposure value of automatic exposure frame
exp_FEF	Exposure value of fixed exposure frame

3.39 TofExpouseCurrentGroup3

Prototype:

```
typedef struct tagTofExpouseCurrentGroup3
{
        UINT32 exp_AEF;//Exposure value of automatic exposure frame
        UINT32 exp_FEF;//Exposure value of fixed exposure frame
        UINT32 exp_Gray;//Exposure value of gray exposure frame
}TofExpouseCurrentGroup3;
```

Description:

Value of TOF exposure (Combination method 3).

Type:

exp_AEF	Exposure value of automatic exposure frame
exp_FEF	Exposure value of fixed exposure frame
exp_Gray	Exposure value of gray exposure frame

3.40 TofExpouseCurrentItems

Prototype:

```
typedef struct tagTofExpouseCurrentItems
{
        UINT32 nIndex;//1---g1 valid, 2---g2 valid

        union
        {
                //[Type 1]: Only applicable when raw data with single-frequency or dual-frequency
                TofExpouseCurrentGroup1 g1;//Exposure value

                //[Type 2]: Only applicable when raw data with automatic exposure frames and fixed
exposure frames（During intra-frame HDRZ fusion）
                TofExpouseCurrentGroup2 g2;//Exposure value
        }uParam;

}TofExpouseCurrentItems;
```

Description:

Combination of options for TOF exposure value.

Type:

nIndex	1---g1 valid, 2---g2 valid
g1	Exposure parameter
g2	Exposure parameter

3.41 TofExpouseRangeGroup1

Prototype:

```
typedef struct tagTofExpouseRangeGroup1
{
	UINT32 min;//Exposure value (minimum)
	UINT32 max;//Exposure value (maximum)
}TofExpouseRangeGroup1;
```

Description:

TOF exposure range（Combination method 1）.

Type:

min	Exposure value (minimum)
max	Exposure value (maximum)

3.42 TofExpouseRangeGroup2

Prototype:

```
typedef struct tagTofExpouseRangeGroup2
{
	UINT32 min_AEF;//Exposure value of automatic exposure frame (minimum)
	UINT32 max_AEF;//Exposure value of automatic exposure frame (maximum)

	UINT32 min_FEF;//Exposure value of fixed exposure frame (minimum)
	UINT32 max_FEF;//Exposure value of fixed exposure frame (maximum)
}TofExpouseRangeGroup2;
```

Description:

TOF exposure range（Combination method 2）.

Type:

min_AEF	Exposure value of automatic exposure frame (minimum)
max_AEF	Exposure value of automatic exposure frame (maximum)
min_FEF	Exposure value of fixed exposure frame (minimum)
max_FEF	Exposure value of fixed exposure frame (maximum)

3.43 TofExpouseRangeGroup3

Prototype:

```
typedef struct tagTofExpouseRangeGroup3
{
    UINT32 min_AEF;//Exposure value of automatic exposure frame (minimum)
    UINT32 max_AEF;//Exposure value of automatic exposure frame (maximum)

    UINT32 min_FEF;//Exposure value of fixed exposure frame (minimum)
    UINT32 max_FEF;//Exposure value of fixed exposure frame (maximum)

    UINT32 min_Gray;//Exposure value of gray exposure frame (minimum)
    UINT32 max_Gray;//Exposure value of gray exposure frame (maximum)
}TofExpouseRangeGroup3;
```

Description:

TOF exposure range（Combination method 3）.

Type:

min_AEF	Exposure value of automatic exposure frame (minimum)
max_AEF	Exposure value of automatic exposure frame (maximum)
min_FEF	Exposure value of fixed exposure frame (minimum)
max_FEF	Exposure value of fixed exposure frame (maximum)
min_Gray	Exposure value of gray exposure frame (minimum)
max_Gray	Exposure value of gray exposure frame (maximum)

3.44 TofExpouseRangeItems

Prototype:

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

    union
    {
        //[Type 1]: Only applicable when raw data with single-frequency or dual-frequency
        TofExpouseRangeGroup1 g1;//Exposure range

        //[Type 2]: Only applicable when raw data with automatic exposure frames and fixed
exposure frames（During intra-frame HDRZ fusion）
        TofExpouseRangeGroup2 g2;//Exposure range

        //[Type 3]: Only applicable when raw data with automatic exposure frames and fixed
exposure frames（During intra-frame HDRZ fusion）, in addition, it is allowed to set gray exposure
frames
        TofExpouseRangeGroup3 g3;//Exposure range
    }uParam;
}TofExpouseRangeItems;
```

Description:

Option combination of TOF exposure range.

Type:

nIndex	1---g1 valid, 2---g2 valid, 3---g3 valid
g1	Exposure range
g2	Exposure range
g3	Exposure range

3.45 CUSTOM_PARAM_GUEST_ID

Prototype:

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

Description:

Customer identification number of custom parameters.

Type:

CUSTOM_PARAM_GUEST_ID_1	Customer 1
CUSTOM_PARAM_GUEST_ID_2	Customer 2

3.46 CustomParamGuest1

Prototype:

```
typedef struct tagCustomParamGuest1
{
        SINT32 quantileThreshold;//AE ratio
        FLOAT32 referenceAmplitude;//Reference amplitude
        FLOAT32 amplitudeThreshold;//Amplitude threshold
        UINT8 szRes[496];//Total 508 bytes, 4 bytes aligned, reserve
}CustomParamGuest1;
```

Description:

Custom parameters of Customer 1.

Type:

quantileThreshold	AE ratio
referenceAmplitude	Reference amplitude
amplitudeThreshold	Amplitude threshold

szRes	byte alignment, reserve

3.47 CustomParamGuest2

Prototype:

```
typedef struct tagCustomParamGuest2
{
	UINT8 szRes[508];//Total 508 bytes, 4 bytes aligned, reserve
}CustomParamGuest2;
```

Description:

Custom parameters of Customer 2.

Type:

szRes	byte alignment, reserve

3.48 GuestCustomParam

Prototype:

```
typedef struct tagGuestCustomParam
{
	CUSTOM_PARAM_GUEST_ID id;//Input parameters, read only

	union
	{
		CustomParamGuest1 p1;//Valid when id is CUSTOM_PARAM_GUEST_ID_1;
		CustomParamGuest2 p2;//Valid when id is CUSTOM_PARAM_GUEST_ID_2;

		UINT8 data[508];//Limit the union to 508 bytes in length（This field is not used, it is only used to define the length of the data structure）
	}uParam;
}GuestCustomParam;
```

Description:

Customer-defined parameter structure.

Type:

id	Customer ID for custom parameters	
uParam	Use different fields according to the id parameter (customer id)	
	p1	Valid when id is CUSTOM_PARAM_GUEST_ID_1
	p2	Valid when id is CUSTOM_PARAM_GUEST_ID_2
	data	Limit the union to 508 bytes in length

3.49 RoiItem

Prototype:

```
typedef struct tagRoiItem
{
    UINT32    left;//Start column, start from 0;
    UINT32    top;//Start row, start from 0;
    UINT32    right;//End column, no more than image width;
    UINT32    bottom;//End row, no more than image height;

}RoiItem;
```

Description:

Structure of ROI region.

Parameter:

left	Start column, start from 0
top	Start row, start from 0
right	End column, no more than image width
bottom	End row, no more than image height

3.50 DepthCalRoi

Prototype:

```
typedef struct tagDepthCalRoi
{
    RoiItem struMax;//Maximum value, read only
    RoiItem struDefault;//Default value, read only

    RoiItem struCurrent;//Current value, readable and writable

}DepthCalRoi;
```

Description:

Structure of ROI region.

Parameter:

struMax	Maximum value, read only
struDefault	Default value, read only
struCurrent	Current value, readable and writable

3.51 TofModuleLensGeneral

Prototype:

```
typedef struct tagTofModuleLensGeneral
{
    FLOAT32 fx;
    FLOAT32 fy;
    FLOAT32 cx;
    FLOAT32 cy;
    FLOAT32 k1;
    FLOAT32 k2;
```

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

Description:

Internal parameters and distortion parameters of TOF module（General model）。

Type:

fx	Parameter fx
fy	Parameter fy
cx	Parameter cx
cy	Parameter cy
k1	Parameter k1
k2	Parameter k2
p1	Parameter p1
p2	Parameter p2
k3	Parameter k3

3.52 TofModuleLensFishEye

Prototype:

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

Description:

Internal parameters and distortion parameters of TOF module（Fisheye model）。

Type:

fx	Parameter fx
fy	Parameter fy
cx	Parameter cx
cy	Parameter cy
k1	Parameter k1
k2	Parameter k2

k3	Parameter k3
k4	Parameter k4

3.53 TofModuleLensParameter

Prototype:

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

Description:

Internal parameters and distortion parameters of TOF module (V1.0 version, it is recommended not to use it again, because it cannot be applied to fisheye models).

Type:

fx	Parameter fx
fy	Parameter fy
cx	Parameter cx
cy	Parameter cy
k1	Parameter k1
k2	Parameter k2
p1	Parameter p1
p2	Parameter p2
k3	Parameter k3

3.54 TofModuleLensParameterV20

Prototype:

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

        union
        {
                //[_Type 1]: General model
                TofModuleLensGeneral general;//General model

                //[_Type 2]: Fisheye model
        TofModuleLensFishEye fishEye;//Fisheye model
    }uParam;

}TofModuleLensParameterV20;
```

Description:

Internal parameters and distortion parameters of TOF module (V2.0 version).

Type:

nIndex		1---general valid, 2---fishEye valid
uParam	general	General model
	fishEye	Fisheye model

3.55 TofCalibData

Prototype:

```
typedef struct tagTofCalibData
{
    UINT8* pData;//Point to calibration data
    UINT32 nDataLen;//Calibration data length in pData
}TofCalibData;
```

Description:

Structure of TOF Module Calibration Data.

Type:

pData	Point to calibration data
nDataLen	Calibration data length in pData

3.56 TofRawData

Prototype:

```
typedef struct tagTofRawData
{
    //RAW data
    UINT8* pRaw;//One frame of RAW data
    UINT32 nRawLen;//RAW data length（byte number）

    //RAW data other attribute parameters
    FLOAT32 fTemperature;//Module temperature when outputting RAW data（Note: Some
model modules do not need this field, and some module RAW data comes with this data, so you
can enter a value of 0）

}TofRawData;
```

Description:

Structure of RAW Data.

Type:

pRaw	One frame of RAW data
nRawLen	RAW data length（byte number）
fTemperature	Module temperature when outputting RAW data （**Note: Some model modules do not need this field, and some module RAW data comes with this data, so you can enter a value of 0**）

3.57 ExterntionHooks

Prototype:

```
typedef struct tagExterntionHooks
{
        void* pUserData;//User-defined data

        /**********Used to send the calculated exposure value in advance **********/
        //@       pExp: Calculated exposure value information;
        //@       user_data: User-defined data, and the same as pUserData;
        //@       [Special attention]: When calling the TOFM_XXX interface in the callback function,
only part of the interface of the software algorithm is allowed to be called, otherwise it will
deadlock！！！！！
        void(*RecvTofExpTime)(TofExpouseCurrentItems* pExp, void*user_data);//Choose
whether to implement according to the actual situation of the module

}ExterntionHooks;
```

Description:

Structure of RAW Data。

Type:

pUserData	User-defined data
RecvTofExpTime	Used to send the calculated exposure value in advance (valid when only output single-frequency or only output dual-frequency raw data at the same time). [Special attention]: When calling the TOFM_XXX interface in the callback function, only part of the interface of the software algorithm is allowed to be called, otherwise it will deadlock!!!! [Explanation of parameters]: pExp: Calculated exposure value information; user_data: User-defined data, and the same as pUserData;

4 Special Data structure and type definition

4.1 TOF_DEV_TYPE

Prototype:

```
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 version）
        TOF_DEV_CLEANER01APRO           = MAKE_UNIQUE_ID('C', 0x01, 'A',
0x02),//Cleaner01A（Plus version）
        TOF_DEV_CLEANER01A_NET          = MAKE_UNIQUE_ID('C', 0x01, 'A',
0x02),//Cleaner01A（Online version）
        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（Online version）
        TOF_DEV_CLEANER01E_NET          = MAKE_UNIQUE_ID('C', 0x01, 'E',
0x01),//Cleaner01E（Online version）
        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（Online version）
        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 version )
        TOF_DEV_MARS05B_BCTC_SUNNY = MAKE_UNIQUE_ID('M', 0x05, 'B',
0x02),//Mars05B(BCTC version _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

        //This part is the demo module
        TOF_DEV_DEMO_3DCP_NET           = MAKE_UNIQUE_ID(0xde, 0x3d, 'C',
0x00),//Demo version 3DCP（Online version）
        TOF_DEV_DEMO_3DCP               = MAKE_UNIQUE_ID(0xde, 0x3d, 'C',
0x01),//Demo version 3DCP
  TOF_DEV_DEMO_C00P01A_NET              = MAKE_UNIQUE_ID(0xC0, 'P', 0x1A,
  0x00),//An RGBD module named C00P01A（Online version）

        TOF_DEV_DEMO_UPG                = MAKE_UNIQUE_ID(0xde,    'U', 'P',    'G'),//demo
version UPG


}TOF_DEV_TYPE;
```

Description:

Model of TOF device.

Type:

TOF_DEV_CHROMEBOOK	ChromeBook
TOF_DEV_CLEANER01A	Cleaner01A
TOF_DEV_CLEANER01APLUS	Cleaner01A(Plus version)
TOF_DEV_CLEANER01APRO	Cleaner01A(Plus version)
TOF_DEV_CLEANER01A_NET	Cleaner01A(Online version)
TOF_DEV_CLEANER01B	Cleaner01B
TOF_DEV_CLEANER01D	Cleaner01D
TOF_DEV_CLEANER01D_NET	Cleaner01D(Online version)
TOF_DEV_CLEANER01E_NET	Cleaner01E(Online version)
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(Online version)
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 version)
TOF_DEV_MARS05B_BCTC_SUNNY	Mars05B (BCTC version _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 version 3DCP(Online version)
TOF_DEV_DEMO_3DCP	Demo version 3DCP
TOF_DEV_DEMO_C00P01A_NET	Demo version An RGBD module named C00P01A（Online version）
TOF_DEV_DEMO_UPG	Demo version UPG

4.2 TofFrameData

Prototype:

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

        //
        FLOAT32* pDepthData;//Radial distance(without filter)
        FLOAT32* pDepthDataFilter;//Radial distance(after filter)

        PointData *pPointData;//point cloud data

        GRAY_FORMAT grayFormat;//Data format in pGrayData
        void      *pGrayData;//Grayscale data

        FLOAT32 *pConfidence;//Confidence（Only for supported boards）

        RgbDData* pRgbD;//RgbD data

        void      *pRawData;//raw data（Only for boards that support raw data）
        UINT32 nRawDataLen;//Raw data length in pRawData, byte number

        //Extended data (Generally targeted at the special needs of customers), Different devices and
different customers are different and may be empty;
        void      *pExtData;//Extended data
        UINT32 nExtDataLen;//Raw data length in pRawData, byte number

}TofFrameData;
```

Description:

Structure of TOF data.

Parameter:

timeStamp	Timestamp of TOF data frame
frameWidth	Width of TOF data frame
frameHeight	Height of TOF data frame
pDepthData	Radial distance(without filter)
pDepthDataFilter	Radial distance(after filter)
pPointData	TOF point cloud data
grayFormat	Data format in pGrayData
pGrayData	TOF IR Image Data
pConfidence	Confidence(Only for supported boards)
pRgbD	RGBD data(Suitable for rgbd support)
pExtData	Extended data (Generally targeted at the special needs of customers), Different devices /different customers are different and it may be empty.
nExtDataLen	Raw data length in pRawData, byte number

4.3 ANALOG_GAIN_MODE

Prototype:

```
typedef enum tagANALOG_GAIN_MODE
{
	ANALOG_GAIN_MODE_MANUAL = 0x00000001,//Manual analog gain
	ANALOG_GAIN_MODE_AUTO     = 0x00000002,//Automatic analog gain
}ANALOG_GAIN_MODE;
```

Description:

Types of TOF analog gain.

Type:

ANALOG_GAIN_MODE_MANUAL	Manual analog gain
ANALOG_GAIN_MODE_AUTO	Automatic analog gain

4.4 DIGITAL_GAIN_MODE

Prototype:

```
typedef enum tagDIGITAL_GAIN_MODE
{
	DIGITAL_GAIN_MODE_MANUAL = 0x00000001,//Manual digital gain
	DIGITAL_GAIN_MODE_AUTO     = 0x00000002,//Automatic digital gain
}DIGITAL_GAIN_MODE;
```

Description:

Types of TOF digital gain.

Type:

DIGITAL_GAIN_MODE_MANUAL	Manual digital gain

DIGITAL_GAIN_MODE_AUTO	Automatic digital gain

4.5 RgbVideoControlProperty

Prototype:

```
typedef enum tagRgbVideoControlProperty
{
	RgbVideoControl_Exposure = 0x00000001,//Exposure attribute of RGB module
	RgbVideoControl_Gain	= 0x00000002,//Gain attribute of RGB module
}RgbVideoControlProperty;
```

Description:

Types of RGB attribute.

Type:

RgbVideoControl_Exposure	Exposure attribute
RgbVideoControl_Gain	Gain attribute

4.6 RgbVideoControlFlags

Prototype:

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

Description:

RGB attribute value (Extended attribute value).

Type:

RgbVideoControlFlags_Auto	Automatic
RgbVideoControlFlags_Manual	Manual

4.7 RgbVideoControl

Prototype:

```
typedef struct tagRgbVideoControl
{
	SINT32		lDefault;//Default value
	SINT32		lStep;//Step value
	SINT32		lMax;//Maximum value
	SINT32		lMin;//Minimum value
	SINT32	lCapsFlags;//Supported value, it's one or more combinations of
RgbVideoControlFlags

	SINT32		lCurrent;//Current value
	RgbVideoControlFlags	lFlags;//Current Flag value

}RgbVideoControl;
```

Description:

RGB attribute value.

Type:

lDefault	Default value
lStep	Step value
lMax	Maximum value
lMin	Minimum value
lCapsFlags	Supported value, it's one or more combinations of RgbVideoControlFlags
lCurrent	Current value
lFlags	Current Flag value, it's one of RgbVideoControlFlags

4.8 RgbFrameData

Prototype:

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

        COLOR_FORMAT formatType;//Point out the format of the data frame in pFrameData
        COLOR_FORMAT formatTypeOrg;//Point out the format of data frame in pFrameData
(Format before encode compression)
        UINT32    nFrameLen;
        UINT8*    pFrameData;

        //Extended data (Generally targeted at the special needs of customers), Different devices and
different customers are different and may be empty;
        void      *pExtData;//Extended data
        UINT32 nExtDataLen;//Length of the extended data in pExtData, byte number

}RgbFrameData;
```

Description:

Structure of the RGB data.

Parameter:

timeStamp	Timestamp of RGB data frame
frameWidth	Width of RGB data frame
frameHeight	Height of RGB data frame
formatType	Point out the format of the data frame in pFrameData
formatTypeOrg	Point out the format of data frame in pFrameData (Encode format before compression)

nFrameLen	Point out the length of data frame in pFrameData
pFrameData	RGB data
pExtData	Extended data (Generally targeted at the special needs of customers), Different devices /different customers are different and may be empty
nExtDataLen	Length of the extended data in pExtData, byte number

4.9 ImuFrameData

Prototype:

```
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;
```

Description:

Structure of IMU data.

Parameter:

timsstamp	IMU timestamp
accelData_x	x axial acceleration
accelData_y	y axial acceleration
accelData_z	z axial acceleration
gyrData_x	x axial acceleration
gyrData_y	y axial acceleration
gyrData_z	z axial acceleration
magData_x	x axial acceleration
magData_y	y axial acceleration
magData_z	z axial acceleration

4.10 TofDevInitParam

Prototype:

typedef struct tagTofDevInitParam

```
{
        SCHAR szDepthCalcCfgFileDir[200];//Directory of the configuration file required for in-depth calculation, such as home/user/temp

        UINT8 nLogLevel;//Log printing level（Not yet effective）

        SBOOL bSupUsb;          //if support USB devices

        SBOOL bSupNetWork; //if support network devices
        SCHAR szHostIPAddr[32];//IP address of a network card on a local host（Can also leave it blank, otherwise all local network cards will be traversed）

        SBOOL bSupSerialCOM;//Whether need to support serial port（Value true when serial port is required）
        SCHAR szSerialDev[64];//A serial port device on a local host（When the bSupSerialCOM field is true, the field is valid）
                                        //Windows environment can leave it blank or fill it in, such as COM1, COM2, et al, if does not fill will traverse all local serial port
                                        //Linux environment must be filled in, such as /dev/ttyS0, /dev/ttyUSB0, et al

        SBOOL bWeakAuthority;//Whether the permission is low (such as Android system without root), which is only applicable to Linux system / Android system

        SBOOL bDisablePixelOffset;//The SDK is disabled to skip some pixel so that the TOF data without address offset(the resolution of TOF data output to the user is the same as raw data)

        SCHAR szLogFile[256]; //A log file that records debug information inside the SDK, for example: home/user/temp/tof_dev_sdk_log.txt

}TofDevInitParam;
```

Description:

Structure of the TOF module SDK initialization parameters.

Parameter:

szDepthCalcCfgFileDir	Directory of the configuration file required for in-depth calculation, such as home/user/temp
nLogLevel	Log printing level（Not yet effective）
bSupUsb	if support USB devices
bSupNetWork;	if support network devices
szHostIPAddr	IP address of a network card on a local host（Can also leave it blank, otherwise all local network cards will be traversed）
bSupSerialCOM	Whether need to support serial port（Value true when serial port is required）
szSerialDev	A serial port device on a local host（When the bSupSerialCOM field is true, the field is valid）; Windows environment can leave it blank or fill it in, such as COM1, COM2, et al, if does not fill will traverse all local serial port ;

	Linux environment must be filled in, such as /dev/ttyS0, /dev/ttyUSB0, et al;
bWeakAuthority	Whether the permission is low (such as Android system without root), which is only applicable to Linux system / Android system
bDisablePixelOffset	The SDK is disabled to skip some pixel so that the TOF data without address offset(the resolution of TOF data output to the user is the same as raw data)
szLogFile	A log file that records debug information inside the SDK, for example: home/user/temp/tof_dev_sdk_log.txt

4.11 TofDeviceDescriptor

Prototype:

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

Description:

Structure of the TOF device description.

Parameter:

hDevice	TOF device handle， it's used in SDK
hDriver	TOF device driver handle， it's used in SDK

4.12 TofDeviceDescriptorWithFd

Prototype:

```
typedef struct tagTofDeviceDescriptorWithFd
{
	SINT32 usbDevFd; //file descriptor of USB device
	UINT16 usbDevVID;//VID of USB device
	UINT16 usbDevPID;//PID of USB device
}TofDeviceDescriptorWithFd;
```

Description:

Structure of the TOF device description(with device file descriptor).

Parameter:

usbDevFd	file descriptor of USB device
usbDevVID	VID of USB device
usbDevPID	PID of USB device

4.13 TofDeviceInfo

Prototype:

```
typedef struct tagTofDeviceInfo
{
        //BASIC information
        TOF_DEV_TYPE devType;//Used to distinguish which device
        SCHAR szDevName[32];
        SCHAR szDevId[64];//Serial number of the device/module（Uniqueness of identification
device）
        SCHAR szFirmwareVersion[32];//Firmware version information

        //TOF
        UINT32 supportedTOFMode;//Combination of TOF _ MODE
        UINT32 tofResWidth;
        UINT32 tofResHeight;
        GRAY_FORMAT grayFormat;//Format of grayscale data

        //TOF Expouse
        UINT32 supportedTofExpMode;//Combination of EXP_MODE
        //TOF Analog Gain
        UINT32 supportedTofAnalogGainMode;//Combination of ANALOG_GAIN_MODE
        //TOF Digital Gain
        UINT32 supportedTofDigitalGainMode;//Combination of DIGITAL_GAIN_MODE

        //TOF Filter
        UINT32 supportedTOFFilter; //Combination of TOF_FILTER

        //TOF HDRZ
        SBOOL bTofHDRZSupported;
        UINT8 byRes1[3];//Byte alignment, reserve

        //TOF RemoveINS
        SBOOL bTofRemoveINSSupported;
        UINT8 byRes5[3];//Byte alignment, reserve

        //TOF MPIFlag
        SBOOL bTofMPIFlagSupported;//[This field has been abolished]
        UINT8 byRes6[3];//Byte alignment, reserve

        //RGB
        SBOOL bRgbSupported;
        UINT8 byRes2[3];//Byte alignment, reserve
        COLOR_FORMAT rgbColorFormat;//Efferent RGB data format
        COLOR_FORMAT rgbColorFormatOrg;//Efferent RGB data format (Format before encode
compression)
        UINT32 rgbResWidth;
        UINT32 rgbResHeight;
        UINT32 supportedRgbProperty;// Combination of RgbVideoControlProperty

        //RGBD
        SBOOL bRgbDSupported;
        UINT8 byRes3[3];//Byte alignment, reserve

        //IMU
        SBOOL bImuSupported;
```

```
    UINT8 byRes4[3];//Byte alignment, reserve

    //Remote capture
    SBOOL bRemoteCaptureSupported;
    //Firmware upgrade
    SBOOL bUpgradeFirmwareSupported;
    //Device restart
    SBOOL bRebootDevSupported;
    //Synchronization time between master and slave machines
    SBOOL bMasterSlaveSyncTimeSupported;

    //

}TofDeviceInfo;
```

Description:

Structure of TOF device information data.

Parameter:

devType	Used to distinguish which device
szDevName	TOF device name, it's for distinguishing module types
szDevId	Serial number of the device/module（Uniqueness of identification device）
szFirmwareVersion	Firmware version information
supportedTOFMode	TOF mode supported by TOF device, which can be various combinations of TOF_MODE
tofResWidth	Width information of TOF resolution
tofResHeight	Height information of TOF resolution
grayFormat	Format of grayscale data
supportedTofExpMode	TOF exposure mode supported by TOF device, which can be various combinations of EXP_MODE
supportedTofAnalogGainMode	TOF analog gain mode supported by TOF device, which can be various combination of ANALOG_GAIN_MODE
supportedTofDigitalGainMode	TOF digital gain mode supported by TOF device, which can be various combinations of DIGITAL_GAIN_MODE
supportedTOFFilter	TOF filter type supported by TOF device, which can be various combinations of TOF_FILTER
bTofHDRZSupported	Whether TOF device support HDRZ output
bTofRemoveINSSupported	Whether TOF device support RemoveINS algorithm
bTofMPIFlagSupported	Whether TOF device support MPIFlag algorithm [This field has been abolished]
bRgbSupported	Whether TOF device support RGB output
rgbColorFormat	Efferent RGB data format

rgbColorFormatOrg	Efferent RGB data format (Format before encode compression)
rgbResWidth	Width information of RGB resolution
rgbResHeight	Height information of RGB resolution
supportedRgbProperty	attribute supported by RGB
bRgbDSupported	Whether TOF device support RGBD output
bImuSupported	TOF device support IMU output
bRemoteCaptureSupported	TOF device support remote capture function （Storage inside the device ）
bUpgradeFirmwareSupported	TOF device support firmware upgrade
bRebootDevSupported	TOF device support restart the device

4.14 TofDeviceParam

Prototype:

```
typedef struct tagTofDeviceParam
{
	FLOAT32 fBoardTemp;//Temperature of the motherboard (Required device support)
	FLOAT32 fSensorTemp;//Temperature of the sensor (Required device support)
	FLOAT32 fImuTemp;//Temperature of the Imu (Required device support)
}TofDeviceParam;
```

Description:

Parameter of device (Generally some dynamically changing read-only parameters).

Type:

fBoardTemp	Temperature of the motherboard (Required device support), (Set to 0.0 generally means not supported)
fSensorTemp	Temperature of the sensor (Required device support), (Set to 0.0 generally means not supported)
fImuTemp	Temperature of the Imu, (Set to 0.0 generally means not supported)

4.15 TofDeviceTemperature

Prototype:

```
typedef struct tagTofDeviceTemperature
{
	FLOAT32 fBoardTemp;//Temperature of the motherboard (Required device support)
	FLOAT32 fSensorTemp;//Temperature of the sensor
	FLOAT32 fImuTemp;//Temperature of the Imu
}TofDeviceTemperature;
```

Description:

Parameters of device temperature information (Different devices obtain different temperature types).

Type:

fBoardTemp	Temperature of the motherboard (Required device support), (Set to 0.0 generally means not supported)
fSensorTemp	Temperature of the sensor (Required device support), (Set to 0.0 generally means not supported)
fImuTemp	Temperature of the Imu, (Set to 0.0 generally means not supported)

4.16 NetDevInfo_t

Prototype:

```
typedef struct tagNetDevInfo
{
	SBOOL bDHCP;//Whether to obtain IP automatically
	UINT8 byRes[3];//Byte alignment, reserve
	SCHAR szIPv4Address[32];//IP address of the device
	SCHAR szIPv4SubnetMask[32];//Subnet mask of the device
	SCHAR szIPv4Gateway[32];//Gateway of the device
	SCHAR szMAC[32];//MAC address of the device

}NetDevInfo_t;
```

Description:

Network information parameters of the device (Only supported by the network access device).

Type:

bDHCP	Whether to obtain IP automatically
szIPv4Address	IP address of the device
szIPv4SubnetMask	Subnet mask of the device
szIPv4Gateway	Gateway of the device
szMAC	MAC address of the device

4.17 RemoteCapture

Prototype:

```
typedef struct tagRemoteCapture
{
		UINT8 szRes[4];//Reserve 4 bytes for alignment
}RemoteCapture;
```

Description:

Remotely control the device to capture pictures and save them inside the device (Supported by some devices).

Type:

szRes	No practical significance, reserved for byte alignment

4.18 FIRMWARE_UPGRADE_STATUS

Prototype:

```
typedef enum tagFIRMWARE_UPGRADE_STATUS
```

```
{
	FIRMWARE_UPGRADE_STATUS_FINISHED     = 1,//Update completed
	FIRMWARE_UPGRADE_STATUS_RUNNING      = 2,//Upgrading
	FIRMWARE_UPGRADE_STATUS_FAILED       = 3,//Upgrade failed
	FIRMWARE_UPGRADE_STATUS_UNKNOWN      = 4,//Upgrade failed (Unknown
error)
	FIRMWARE_UPGRADE_STATUS_ERROR_DATA = 5,//Upgrade failed (Firmware
package error)
	FIRMWARE_UPGRADE_STATUS_IO           = 6,//Upgrade failed (IO read and write
failed)
}FIRMWARE_UPGRADE_STATUS;
```

Description:

Real-time status of firmware upgrade.

Type:

FIRMWARE_UPGRADE_STATUS_FINISHED	Update completed
FIRMWARE_UPGRADE_STATUS_RUNNING	Upgrading
FIRMWARE_UPGRADE_STATUS_FAILED	Upgrade failed
FIRMWARE_UPGRADE_STATUS_UNKNOWN	Upgrade failed (Unknown error)
FIRMWARE_UPGRADE_STATUS_ERROR_DATA	Upgrade failed (Firmware package error)
FIRMWARE_UPGRADE_STATUS_IO	Upgrade failed (IO read and write failed)

4.19 FirmwareUpgradeStatus

Prototype:

```
typedef struct tagFirmwareUpgradeStatus
{
	FIRMWARE_UPGRADE_STATUS status;//Upgrade status, refer to value
FIRMWARE_UPGRADE_STATUS
	UINT8 nProgress;//Real-time progress, the value must be between 0 and 100
	UINT8 byRes[3];//Byte alignment, reserve
}FirmwareUpgradeStatus;
```

Description:

Real-time status information of firmware upgrade.

Type:

status	Upgrade status
nProgress	Real-time progress, the value must be between 0 and 100
byRes	Byte alignment, reserve

4.20 FNFirmwareUpgradeStatus

Prototype:

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

Description:

Callback function of the real-time status of the firmware upgrade.

Parameter:

statusData	Real-time status information of firmware upgrade
pUserData	User data pointer

4.21 FirmwareUpgradeData

Prototype:

```
typedef struct tagFirmwareUpgradeData
{
	UINT8* pData;//Point to the firmware data (First address of the complete firmware data)
	UINT32 nDataLen;//Length of the firmware data in pData (Length of the complete firmware data)

	FNFirmwareUpgradeStatus fnUpgradeStatus;//Callback function of the real-time status of the firmware upgrade
	void* pUpgradeStatusUserData;//fnUpgradeStatus的pUserDataParameter
}FirmwareUpgradeData;
```

Description:

Firmware package data.

Type:

pData	Point to the firmware data (First address of the complete firmware data)
nDataLen	Length of the firmware data in pData (Length of the complete firmware data)
fnUpgradeStatus	Callback function of the real-time status of the firmware upgrade
pUpgradeStatusUserData	pUserData parameter of fnUpgradeStatus

4.22 RebootDev

Prototype:

```
typedef struct tagRebootDev
{
	UINT8 byRes[4];//Byte alignment, reserve
}RebootDev;
```

Description:

Device restart.

Type:

byRes	Byte alignment, reserve

4.23 MasterSlaveSyncTime

Prototype:

```
typedef struct tagMasterSlaveSyncTime
{
	UINT64 hostSendTimestamp;//Time when the host sent the command (Local time of the host)
	UINT64 slaveRecvTimestamp;//Time when the slaver received the command (Local time of the slaver)
	UINT64 slaveSendTimestamp;//Time when the slaver sent the command (Local time of the slaver)
	UINT64 hostRecvTimestamp;//Time when the host received the command (Local time of the host)

}MasterSlaveSyncTime;
```

Description:

Parameters of binocular camera.

Type:

hostSendTimestamp	Time when the host sent the command (Local time of the host)
slaveRecvTimestamp	Time when the slaver received the command (Local time of the slaver)
slaveSendTimestamp	Time when the slaver sent the command (Local time of the slaver)
hostRecvTimestamp	Time when the host received the command (Local time of the host)

4.24 TofAnalogGain

Prototype:

```
typedef struct tagTofAnalogGain
{
	SBOOL	bAuto;//Whether automatic
	UINT8	szRes[2];//4-byte alignment, reserve
	SBOOL	bUpdataValue;//Whether to update the gain value to the board (This field is only valid when setting)
	SINT32 lCurrent;//Current value

	SINT32 lDefault;//Default value (This field is only valid at the time of acquisition)
	SINT32 lStep;//Step value (This field is only valid at the time of acquisition)
	SINT32 lMax;//Maximum value (This field is only valid at the time of acquisition)
	SINT32 lMin;//Minimum value (This field is only valid at the time of acquisition)
}TofAnalogGain;
```

Description:

TOF analog gain.

Type:

bAuto	Whether automatic
szRes	4-byte alignment, reserve
bUpdataValue	Whether to update the gain value to the board (This field is only valid when setting)
lCurrent	Current value
lDefault	Default value (This field is only valid at the time of acquisition)
lStep	Step value (This field is only valid at the time of acquisition)
lMax	Maximum value (This field is only valid at the time of acquisition)
lMin	Minimum value (This field is only valid at the time of acquisition)

4.25 TofDigitalGain

Prototype:

```
typedef struct tagTofDigitalGain
{
        SBOOL    bAuto;//Whether automatic
        UINT8    szRes[2];//4-byte alignment, reserve
        SBOOL    bUpdataValue;//Whether to update the gain value to the board (This field is only
valid when setting)
        SINT32 lCurrent;//Current value

        SINT32 lDefault;//Default value (This field is only valid at the time of acquisition)
        SINT32 lStep;//Step value (This field is only valid at the time of acquisition)
        SINT32 lMax;//Maximum value (This field is only valid at the time of acquisition)
        SINT32 lMin;//Minimum value (This field is only valid at the time of acquisition)
}TofDigitalGain;
```

Description:

TOF digital gain.

Type:

bAuto	Whether automatic
szRes	4-byte alignment, reserve
bUpdataValue	Whether to update the gain value to the board (This field is only valid when setting)
lCurrent	Current value
lDefault	Default value (This field is only valid at the time of acquisition)
lStep	Step value (This field is only valid at the time of acquisition)

lMax	Maximum value (This field is only valid at the time of acquisition)
lMin	Minimum value (This field is only valid at the time of acquisition)

4.26 TofFrameDataPixelOffset

Prototype:

```
typedef struct tagTofFrameDataPixelOffset
{
	UINT32 nOffset;//pixel offsets (pixel cnt)

}TofFrameDataPixelOffset;
```

Description:

The number of pixel offsets of TOF data (obtained from TOF callback function) relative to raw data.

Type:

nOffset	pixel offsets (pixel cnt)

4.27 TofSensorStatus

Prototype:

```
typedef enum tagTofSensorStatus
{
	TofSensorStatus_StreamOff = 1,//Sensor is stream off
	TofSensorStatus_StreamOn  = 2,//Sensor is stream on

}TofSensorStatus;
```

Description:

TOF Sensor status.

Type:

TofSensorStatus_StreamOff	Sensor is stream off
TofSensorStatus_StreamOn	Sensor is stream on

4.28 TofSensorStatusCtrl

Prototype:

```
typedef struct tagTofSensorStatusCtrl
{
	TofSensorStatus status;

}TofSensorStatusCtrl;
```

Description:

TOF sensor status control parameter.

Type:

status	TOF Sensor status

4.29 RgbSensorStatusCtrl

Prototype:

```
typedef struct tagRgbfSensorStatusCtrl
{
      UINT8 szRes[4];//reserved

}RgbSensorStatusCtrl;
```

Description:

RGB sensor status control parameter.

Type:

szRes	reserved

4.30 SensorStatusCtrl

Prototype:

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

      union
      {
            TofSensorStatusCtrl struTof;//TOF Sensor status control parameter

            RgbSensorStatusCtrl struRgb;//RGB Sensor status control parameter

      }uParam;

}SensorStatusCtrl;
```

Description:

sensor status control.

Type:

nIndex	1---struTof is valid, 2---struRgb is valid
struTof	TOF Sensor status control parameter
struRgb	RGB Sensor status control parameter

4.31 TOF_DEV_PARAM_TYPE

Prototype:

```
typedef enum tagTOF_DEV_PARAM_TYPE
{
	TOF_DEV_PARAM_Temperature                = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x00),//Temperature information
	TOF_DEV_PARAM_TofLensParameter           = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x01),//Internal parameters and distortion parameters of the TOF module (V1.0 version, it’s recommended not to use it again, because it cannot be applied to the fisheye model)
	TOF_DEV_PARAM_TofCalibData               = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x02),//Calibration data of the TOF module
	TOF_DEV_PARAM_netdevinfo                 = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x03),// Network access device information
	TOF_DEV_PARAM_ReplaceTofCalibData        = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x04),//Replace the calibration data of the TOF module in the SDK (It is just to replace the calibration data in the SDK, not to write to the module)
	TOF_DEV_PARAM_RemoteCapture              = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x05), //Remote capture: Control module capture data and save it inside the module;
	TOF_DEV_PARAM_ExportRaw                  = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x06), //Export one frame of RAW data: Export one frame of RAW data from the module in real time (Suitable for asynchronous transmission of RAW data and depth data);
	TOF_DEV_PARAM_RgbLensParameter           = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x07),//Internal parameters and distortion parameters of the RGB module
	TOF_DEV_PARAM_UpgradeFirmware            = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x08),//Upgrade firmware
	TOF_DEV_PARAM_RebootDev                  = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x09),// Device restart
	TOF_DEV_PARAM_StereoLensParameter        = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x0a),//Parameters of binocular camera
	TOF_DEV_PARAM_GetMasterSlaveSyncTime = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x0b),//Get synchronization time between master and slave machine
	TOF_DEV_PARAM_TofAnalogGain              = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x0c),//TOF analog gain
	TOF_DEV_PARAM_TofDigitalGain             = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x0d),//TOF digital gain
	TOF_DEV_PARAM_TofLensParameterV20        = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x0e),//Internal parameters and distortion parameters of the TOF module (V2.0 version)
	TOF_DEV_PARAM_TofFrameDataPixelOffset= MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x0f),//The number of pixel offsets of TOF data (obtained from TOF callback function) relative to raw data
	TOF_DEV_PARAM_DepthCalRoi                = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x10),//Area for depth calculation
	TOF_DEV_PARAM_SensorStatusCtrl           = MAKE_UNIQUE_ID(0x00, 0x00, 0x00, 0x11),//Sensor status control

}TOF_DEV_PARAM_TYPE;
```

Description:

Type of device parameters.

Type:

TOF_DEV_PARAM_Temperature	Temperature information

TOF_DEV_PARAM_TofLensParameter	Internal parameters and distortion parameters of the TOF module (V1.0 version, it's recommended not to use it again, because it cannot be applied to the fisheye model)
TOF_DEV_PARAM_RgbLensParameter	Internal parameters and distortion parameters of the RGB module
TOF_DEV_PARAM_TofCalibData	Calibration data of the TOF module
TOF_DEV_PARAM_netdevinfo	Network access device information
TOF_DEV_PARAM_ReplaceTofCalibData	Replace the calibration data of the TOF module in the SDK (It is just to replace the calibration data in the SDK, not to write to the module)
TOF_DEV_PARAM_RemoteCapture	Remote capture: Control module capture data and save it inside the module
TOF_DEV_PARAM_ExportRaw	Export one frame of RAW data: Export one frame of RAW data from the module in real time (Suitable for asynchronous transmission of RAW data and depth data)
TOF_DEV_PARAM_UpgradeFirmware	Upgrade firmware
TOF_DEV_PARAM_RebootDev	Device restart
TOF_DEV_PARAM_StereoLensParameter	Parameters of binocular camera
TOF_DEV_PARAM_GetMasterSlaveSyncTime	Get synchronization time between master and slave machine
TOF_DEV_PARAM_TofAnalogGain	TOF analog gain
TOF_DEV_PARAM_TofDigitalGain	TOF digital gain
TOF_DEV_PARAM_TofLensParameterV20	Internal parameters and distortion parameters of the TOF module (V2.0 version)
TOF_DEV_PARAM_TofFrameDataPixelOffset	The number of pixel offsets of TOF data (obtained from TOF callback function) relative to raw data
TOF_DEV_PARAM_DepthCalRoi	Area for depth calculation
TOF_DEV_PARAM_SensorStatusCtrl	Sensor status control

4.32 TofDeviceParamV20

Prototype:

```
typedef struct tagTofDeviceParamV20
{
    TOF_DEV_PARAM_TYPE type;//Input parameters, read only

    union
    {
        TofDeviceTemperature    struTemperature;//Temperature information[Valid when type is TOF_DEV_PARAM_Temperature]
        TofModuleLensParameter struTofLensParameter;//Internal parameters and distortion parameters of the TOF module[Valid when type is TOF_DEV_PARAM_TofLensParameter](V1.0 version, it is recommended not to use it again, because it cannot be applied to the fisheye model)
        TofModuleLensParameterV20 struTofLensParameterV20;//Internal parameters and
distortion parameters of the TOF module[Valid when type is TOF_DEV_PARAM_TofLensParameterV20](V2.0 version)
		RgbModuleLensParameter struRgbLensParameter;//Internal parameters and distortion parameters of the RGB module[Valid when type is TOF_DEV_PARAM_RgbLensParameter]
		TofCalibData			struTofCalibData;//Calibration data of TOF module[Valid when type is TOF_DEV_PARAM_TofCalibData]
		NetDevInfo_t			stuNetDevData;//Network access device information[Valid when type is TOF_DEV_PARAM_netdevinfo]
		TofCalibData			struReplaceTofCalibData;//Replace the calibration data of the TOF module in the SDK[Valid when type is TOF_DEV_PARAM_ReplaceTofCalibData]
		RemoteCapture			struRemoteCapture;//Remote capture: Control module capture data and save it inside the module；[Valid when type is TOF_DEV_PARAM_RemoteCapture]
		TofRawData			struExportRaw;//Export one frame of RAW data: Export one frame of RAW data from the module in real time (Suitable for asynchronous transmission of RAW data and depth data)；[Valid when type is TOF_DEV_PARAM_ExportRaw]
		FirmwareUpgradeData	struFirmware;//Firmware upgrade data[Valid when type is TOF_DEV_PARAM_UpgradeFirmware]
		RebootDev			struRebootDev;//Device restart[Valid when type is TOF_DEV_PARAM_RebootDev]
		StereoLensParameter	struStereoLensParameter;//Parameters of binocular camera[Valid when type is TOF_DEV_PARAM_StereoLensParameter]
		MasterSlaveSyncTime	struMasterSlaveSyncTime;//Synchronization time between master and slave machines[Valid when type is TOF_DEV_PARAM_GetMasterSlaveSyncTime]
		TofAnalogGain			struTofAnalogGain;//TOF analog gain[Valid when type is TOF_DEV_PARAM_TofAnalogGain]
		TofDigitalGain			struTofDigitalGain;//TOF digital gain[Valid when type is TOF_DEV_PARAM_TofDigitalGain]
		TofFrameDataPixelOffset struPixelOffset;//The number of pixel offsets of TOF data (obtained from TOF callback function) relative to raw data[Valid when type is TOF_DEV_PARAM_TofFrameDataPixelOffset]
		DepthCalRoi			struDepthCalRoi;//Area for depth calculation[Valid when type is TOF_DEV_PARAM_DepthCalRoi]
		SensorStatusCtrl		struSensorStatusCtrl;//Sensor status control[Valid when type is TOF_DEV_PARAM_SensorStatusCtrl]

	}uParam;
}TofDeviceParamV20;
```

Description：

Device parameters (Data structure of 2.0 version).

Type：

type	Specified device parameter type, input parameter, read-only
struTemperature	Temperature information[Valid when type is TOF_DEV_PARAM_Temperature]
struTofLensParameter	Internal parameters and distortion parameters of the TOF module[Valid when type is TOF_DEV_PARAM_TofLensParameter](V1.0 version, it is recommended not to use it again, because it cannot be applied to the fisheye model)

struTofLensParameterV20	Internal parameters and distortion parameters of the TOF module[Valid when type is TOF_DEV_PARAM_TofLensParameterV20](V2.0 version)
struRgbLensParameter	Internal parameters and distortion parameters of the RGB module[Valid when type is TOF_DEV_PARAM_RgbLensParameter]
struTofCalibData	Calibration data of TOF module[Valid when type is TOF_DEV_PARAM_TofCalibData]
stuNetDevData	Network access device information[Valid when type is TOF_DEV_PARAM_netdevinfo]
struReplaceTofCalibData	Replace the calibration data of the TOF module in the SDK[Valid when type is TOF_DEV_PARAM_ReplaceTofCalibData]
struRemoteCapture	Remote capture: Control module capture data and save it inside the module; [Valid when type is TOF_DEV_PARAM_RemoteCapture]
struExportRaw	Export one frame of RAW data: Export one frame of RAW data from the module in real time (Suitable for asynchronous transmission of RAW data and depth data)；[Valid when type is TOF_DEV_PARAM_ExportRaw]
struFirmware	Firmware upgrade data[Valid when type is TOF_DEV_PARAM_UpgradeFirmware]
struRebootDev	Device restart[Valid when type is TOF_DEV_PARAM_RebootDev]
struStereoLensParameter	Parameters of binocular camera[Valid when type is TOF_DEV_PARAM_StereoLensParameter]
struMasterSlaveSyncTime	Synchronization time between master and slave machines[Valid when type is TOF_DEV_PARAM_GetMasterSlaveSyncTime]
struTofAnalogGain	TOF analog gain[Valid when type is TOF_DEV_PARAM_TofAnalogGain]
struTofDigitalGain	TOF digital gain[Valid when type is TOF_DEV_PARAM_TofDigitalGain]
struPixelOffset	The number of pixel offsets of TOF data (obtained from TOF callback function) relative to raw data[Valid when type is TOF_DEV_PARAM_TofFrameDataPixelOffset]
struDepthCalRoi	Area for depth calculation[Valid when type is TOF_DEV_PARAM_DepthCalRoi]
struSensorStatusCtrl	Sensor status control[Valid when type is TOF_DEV_PARAM_SensorStatusCtrl]

4.33 TOFDEV_STATUS

Prototype:

typedef enum tagTOFDEV_STATUS
{
TOFDEV_STATUS_UNUSED = MAKE_UNIQUE_ID('U', 'U', 'S', 'E'),//(This value is not used, and the valid device status starts from 1)

```
        TOFDEV_STATUS_DEV_BROKEN                    = MAKE_UNIQUE_ID('D', 'E', 'V',
'B'),//_Device disconnected abnormally
        //
        TOFDEV_STATUS_READ_CALIB_DATA_SUC       = MAKE_UNIQUE_ID('R', 'C', 'D',
'S'),//Successful to read calibration data
        TOFDEV_STATUS_READ_CALIB_DATA_FAILED = MAKE_UNIQUE_ID('R', 'C', 'D',
'F'),//Failed to read calibration data
        //
        TOFDEV_STATUS_TOF_STREAM_FAILED          = MAKE_UNIQUE_ID('T', 'S', 'F',
0x00),//Failed to get TOF stream

}TOFDEV_STATUS;
```

Description:

TOF device status, the device may be in the union of TOF, RGB, RGBD, IMU data stream open status.

Parameter:

TOFDEV_STATUS_UNUSED	This value is not used, and the valid device status starts from 1
TOFDEV_STATUS_DEV_BROKEN	Device disconnected abnormally
TOFDEV_STATUS_READ_CALIB_DATA_SUC	Successful to read calibration data
TOFDEV_STATUS_READ_CALIB_DATA_FAILED	Failed to read calibration data
TOFDEV_STATUS_TOF_STREAM_FAILED	Failed to get TOF stream

4.34 HTOFD

Prototype:
typedef void* HTOFD;

Description:

TOF device handle, this handle points to the memory area managed by the TOF SDK internal device.

4.35 FNTofStream

Prototype:

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

Description:

Callback function for TOF output point cloud and IR image data.

Parameter:

tofFrameData	Structure Pointer for TOF Point Cloud and IR Image Data, refer to TofFrameData in this chapter
pUserData	User data pointer, which is the same as pUserData of TOFD_StartTofStream.

4.36 FNTofDeviceStatus

Prototype:

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

Description:

Callback function of TOF device status.

Parameter:

tofDevStatus	TOF device status, refer to TOFDEV_STATUS in this chapter
pUserData	User data pointer, which is the same as pUserData of TOFD_OpenDevice

4.37 FNRgbStream

Prototype:

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

Description:

Callback function for TOF output RGB image data.

Parameter:

rgbFrameData	RGB data structure pointer, refer to RgbFrameData in this chapter
pUserData	User data pointer, which is the same as pUserData of TOFD_StartRgbStream

4.38 FNImuStream

Prototype:

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

Description:

Callback function for TOF output IMU data.

Parameter:

imuFrameData	IMU data structure pointer, refer to ImuFrameData in this chapter
pUserData	User data pointer, which is the same as pUserData of TOFD_StartImuStream

5 Sample code

See the demo source file in the SDK package for details.