TYCamport3

3

Generated by Doxygen 1.8.17

1	Main Page	1
	1.1 Note	1
2	Class Index	3
	2.1 Class List	3
3	File Index	5
	3.1 File List	5
4	Class Documentation	7
	4.1 DepthEnhenceParameters Struct Reference	7
	4.1.1 Detailed Description	7
	4.2 DepthSpeckleFilterParameters Struct Reference	7
	4.2.1 Detailed Description	7
	4.3 pattern_bin_param Struct Reference	8
	4.3.1 Detailed Description	8
	4.4 pattern_gray_param Struct Reference	8
	4.4.1 Detailed Description	8
	4.5 pattern_sine_param Struct Reference	8
	4.5.1 Detailed Description	8
	4.6 TY_PHC_GROUP_ATTR::phc_group_attr Struct Reference	9
	4.6.1 Detailed Description	9
	4.7 TY_ACC_BIAS Struct Reference	9
	4.7.1 Detailed Description	9
	4.8 TY_ACC_MISALIGNMENT Struct Reference	9
	4.8.1 Detailed Description	10
	4.9 TY_ACC_SCALE Struct Reference	10
	4.9.1 Detailed Description	10
	4.10 TY_AEC_ROI_PARAM Struct Reference	11
	4.10.1 Detailed Description	11
	4.11 TY_BYTEARRAY_ATTR Struct Reference	11
	4.11.1 Detailed Description	11
	4.11.2 Member Data Documentation	11
	4.11.2.1 unit size	12
	4.11.2.2 valid_size	12
	4.12 TY_CAMERA_CALIB_INFO Struct Reference	12
	4.12.1 Detailed Description	13
	4.13 TY_CAMERA_DISTORTION Struct Reference	13
	4.13.1 Detailed Description	13
	4.14 TY_CAMERA_EXTRINSIC Struct Reference	13
	4.14.1 Detailed Description	13
	4.15 TY_CAMERA_INTRINSIC Struct Reference	14
	4.15.1 Detailed Description	14

4.16 TY_CAMERA_ROTATION Struct Reference	15
4.16.1 Detailed Description	15
4.17 TY_CAMERA_STATISTICS Struct Reference	15
4.17.1 Detailed Description	15
4.18 TY_CAMERA_TO_IMU Struct Reference	16
4.18.1 Detailed Description	16
4.19 TY_DEVICE_BASE_INFO Struct Reference	16
4.19.1 Detailed Description	17
4.20 TY_DEVICE_NET_INFO Struct Reference	17
4.20.1 Detailed Description	18
4.21 TY_DEVICE_USB_INFO Struct Reference	18
4.21.1 Detailed Description	18
4.22 TY_DI_WORKMODE Struct Reference	18
4.22.1 Detailed Description	18
4.23 TY_DO_WORKMODE Struct Reference	19
4.23.1 Detailed Description	19
4.24 TY_ENUM_ENTRY Struct Reference	19
4.24.1 Detailed Description	19
4.25 TY_EVENT_INFO Struct Reference	20
4.25.1 Detailed Description	20
4.26 TY_FEATURE_INFO Struct Reference	20
4.26.1 Detailed Description	20
4.27 TY_FLOAT_RANGE Struct Reference	21
4.27.1 Detailed Description	21
4.28 TY_FRAME_DATA Struct Reference	21
4.28.1 Detailed Description	22
4.29 TY_GYRO_BIAS Struct Reference	22
4.29.1 Detailed Description	22
4.30 TY_GYRO_MISALIGNMENT Struct Reference	22
4.30.1 Detailed Description	23
4.31 TY_GYRO_SCALE Struct Reference	23
4.31.1 Detailed Description	23
4.32 TY_IMAGE_DATA Struct Reference	24
4.32.1 Detailed Description	24
4.33 TY_IMU_DATA Struct Reference	24
4.33.1 Detailed Description	25
4.34 TY_INT_RANGE Struct Reference	25
4.34.1 Detailed Description	25
4.35 TY_INTERFACE_INFO Struct Reference	25
4.35.1 Detailed Description	26
4.36 TY_ISP_FEATURE_INFO Struct Reference	26
4.36.1 Detailed Description	26

4.37 TY_LASER_PARAM Struct Reference	2
4.37.1 Detailed Description	2
4.38 TY_LASER_PATTERN_PARAM Struct Reference	2
4.38.1 Detailed Description	2
4.39 TY_PHC_GROUP_ATTR Struct Reference	2
4.39.1 Detailed Description	2
4.40 TY_PIXEL_COLOR_DESC Struct Reference	2
4.40.1 Detailed Description	2
4.41 TY_PIXEL_DESC Struct Reference	2
4.41.1 Detailed Description	2
4.42 TY_TEMP_DATA Struct Reference	2
4.42.1 Detailed Description	2
4.43 TY_TOF_FREQ Struct Reference	3
4.43.1 Detailed Description	3
4.44 TY_TRIGGER_PARAM Struct Reference	3
4.44.1 Detailed Description	3
4.45 TY_TRIGGER_PARAM_EX Struct Reference	3
4.45.1 Detailed Description	3
4.46 TY_TRIGGER_TIMER_LIST Struct Reference	3
4.46.1 Detailed Description	3
4.47 TY_TRIGGER_TIMER_PERIOD Struct Reference	3
4.47.1 Detailed Description	3
4.48 TY_VECT_3F Struct Reference	3
4.48.1 Detailed Description	3
4.49 TY_VERSION_INFO Struct Reference	3
4.49.1 Detailed Description	3
5 File Documentation	3
5.1 TYApi.h File Reference	
5.1.1 Detailed Description	
5.1.2 Function Documentation	
5.1.2.1 TYAppendLogToFile()	
5.1.2.2 TYAppendLogToServer()	
5.1.2.3 TYClearBufferQueue()	
5.1.2.4 TYCloseDevice()	3
5.1.2.5 TYCloseInterface()	4
5.1.2.6 TYDeinitLib()	4
5.1.2.7 TYDisableComponents()	4
5.1.2.8 TYEnableComponents()	
5.1.2.9 TYEnqueueBuffer()	
5.1.2.10 TYErrorString()	4
5.1.2.11 TYFetchFrame()	4

5.1.2.12 TYForceDeviceIP()	45
5.1.2.13 TYGetBool()	47
5.1.2.14 TYGetByteArray()	48
5.1.2.15 TYGetByteArrayAttr()	49
5.1.2.16 TYGetByteArraySize()	51
5.1.2.17 TYGetComponentIDs()	52
5.1.2.18 TYGetDeviceFeatureInfo()	53
5.1.2.19 TYGetDeviceFeatureNumber()	54
5.1.2.20 TYGetDeviceInfo()	55
5.1.2.21 TYGetDeviceInterface()	56
5.1.2.22 TYGetDeviceList()	57
5.1.2.23 TYGetDeviceNumber()	58
5.1.2.24 TYGetDeviceXML()	58
5.1.2.25 TYGetDeviceXMLSize()	59
5.1.2.26 TYGetEnabledComponents()	60
5.1.2.27 TYGetEnum()	61
5.1.2.28 TYGetEnumEntryCount()	62
5.1.2.29 TYGetEnumEntryInfo()	63
5.1.2.30 TYGetFeatureInfo()	65
5.1.2.31 TYGetFloat()	66
5.1.2.32 TYGetFloatRange()	67
5.1.2.33 TYGetFrameBufferSize()	68
5.1.2.34 TYGetInt()	69
5.1.2.35 TYGetInterfaceList()	71
5.1.2.36 TYGetInterfaceNumber()	71
5.1.2.37 TYGetIntRange()	72
5.1.2.38 TYGetString()	73
5.1.2.39 TYGetStringLength()	74
5.1.2.40 TYGetStruct()	77
5.1.2.41 TYHasDevice()	78
5.1.2.42 TYHasFeature()	79
5.1.2.43 TYHasInterface()	80
5.1.2.44 TYLibVersion()	81
5.1.2.45 TYOpenDevice()	81
5.1.2.46 TYOpenDeviceWithIP()	83
5.1.2.47 TYOpenInterface()	84
5.1.2.48 TYRegisterEventCallback()	85
5.1.2.49 TYRegisterImuCallback()	85
5.1.2.50 TYRemoveLogFile()	86
5.1.2.51 TYRemoveLogServer()	86
5.1.2.52 TYSendSoftTrigger()	87
5.1.2.53 TYSetBool()	88

5.1.2.54 TYSetByteArray()	00
5.1.2.55 TYSetEnum()	
5.1.2.56 TYSetFloat()	
5.1.2.57 TYSetInt()	
5.1.2.58 TYSetLogLevel()	
5.1.2.59 TYSetLogPrefix()	
5.1.2.60 TYSetString()	
5.1.2.61 TYSetStruct()	
5.1.2.62 TYStartCapture()	
5.1.2.63 TYStopCapture()	
5.1.2.64 TYUpdateAllDeviceList()	
5.1.2.65 TYUpdateDeviceList()	
5.1.2.66 TYUpdateInterfaceList()	
5.2 TYCoordinateMapper.h File Reference	
5.2.1 Detailed Description	
5.2.2 Macro Definition Documentation	
5.2.2.1 TYMAP_CHECKRET	
5.2.3 Function Documentation	
5.2.3.1 TYDepthImageFillEmptyRegion()	
5.2.3.2 TYInvertExtrinsic()	
5.2.3.3 TYMapDepthImageToPoint3d()	
5.2.3.4 TYMapDepthToPoint3d()	
5.2.3.5 TYMapPoint3dToDepth()	107
5.2.3.6 TYMapPoint3dToDepthImage()	107
5.2.3.7 TYMapPoint3dToPoint3d()	108
5.3 TYDefs.h File Reference	108
5.3.1 Detailed Description	118
5.3.2 Macro Definition Documentation	118
5.3.2.1 TY_DECLARE_IMAGE_MODE1	118
5.3.3 Typedef Documentation	119
5.3.3.1 TY_ACC_BIAS	119
5.3.3.2 TY_ACC_MISALIGNMENT	119
5.3.3.3 TY_ACC_SCALE	119
5.3.3.4 TY_ACCESS_MODE_LIST	119
5.3.3.5 TY_BYTEARRAY_ATTR	120
5.3.3.6 TY_CAMERA_CALIB_INFO	120
5.3.3.7 TY_CAMERA_DISTORTION	120
5.3.3.8 TY_CAMERA_EXTRINSIC	120
5.3.3.9 TY_CAMERA_INTRINSIC	121
5.3.3.10 TY_CAMERA_ROTATION	121
5.3.3.11 TY_CAMERA_TO_IMU	122
5.3.3.12 TY_COMPONENT_ID	122

5.3.3.13 TY_DEVICE_BASE_INFO	122
5.3.3.14 TY_DEVICE_COMPONENT_LIST	122
5.3.3.15 TY_ENUM_ENTRY	123
5.3.3.16 TY_FEATURE_ID	123
5.3.3.17 TY_FLOAT_RANGE	123
5.3.3.18 TY_GYRO_BIAS	123
5.3.3.19 TY_GYRO_MISALIGNMENT	124
5.3.3.20 TY_GYRO_SCALE	124
5.3.3.21 TY_INTERFACE_INFO	124
5.3.3.22 TY_INTERFACE_TYPE_LIST	124
5.3.3.23 TY_PIXEL_BITS_LIST	125
5.3.3.24 TY_TRIGGER_MODE_LIST	125
5.3.4 Enumeration Type Documentation	125
5.3.4.1 TY_ACCESS_MODE_LIST	125
5.3.4.2 TY_DEVICE_COMPONENT_LIST	125
5.3.4.3 TY_FEATURE_ID_LIST	126
5.3.4.4 TY_INTERFACE_TYPE_LIST	129
5.3.4.5 TY_PIXEL_BITS_LIST	129
5.3.4.6 TY_PIXEL_FORMAT_LIST	129
5.3.4.7 TY_RESOLUTION_MODE_LIST	130
5.3.4.8 TY_TRIGGER_MODE_LIST	131
5.4 TYImageProc.h File Reference	131
5.4.1 Detailed Description	133
5.4.2 Function Documentation	133
5.4.2.1 TYDepthEnhenceFilter()	133
5.4.2.2 TYDepthSpeckleFilter()	133
5.4.2.3 TYImageProcesAcceEnable()	134
5.4.2.4 TYUndistortImage()	134
5.5 Tylsp.h File Reference	135
5.5.1 Detailed Description	137
5.5.2 Enumeration Type Documentation	137
5.5.2.1 TY_ISP_FEATURE_ID	137
Index	139

# **Chapter 1**

# Main Page

Copyright(C)2016-2023 Percipio All Rights Reserved

#### **1.1 Note**

Depth camera, called "device", consists of several components. Each component is a hardware module or virtual module, such as RGB sensor, depth sensor. Each component has its own features, such as image width, exposure time, etc..

NOTE: The component TY\_COMPONENT\_DEVICE is a virtual component that contains all features related to the whole device, such as trigger mode, device IP.

Each frame consists of several images. Normally, all the images have identical timestamp, means they are captured at the same time.

2 Main Page

# Chapter 2

# **Class Index**

# 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

DepthEnhenceParameters
DepthSpeckleFilterParameters
pattern_bin_param
pattern_gray_param
pattern_sine_param
TY_PHC_GROUP_ATTR::phc_group_attr
TY_ACC_BIAS
TY_ACC_MISALIGNMENT
TY_ACC_SCALE
TY_AEC_ROI_PARAM
TY_BYTEARRAY_ATTR
Byte array data structure
TY_CAMERA_CALIB_INFO 12
TY_CAMERA_DISTORTION
TY_CAMERA_EXTRINSIC
TY_CAMERA_INTRINSIC 14
TY_CAMERA_ROTATION
TY_CAMERA_STATISTICS
TY_CAMERA_TO_IMU
TY_DEVICE_BASE_INFO
TY_DEVICE_NET_INFO
Device network information
TY_DEVICE_USB_INFO
TY_DI_WORKMODE
TY_DO_WORKMODE
TY_ENUM_ENTRY
TY_EVENT_INFO
TY_FEATURE_INFO
TY_FLOAT_RANGE
Float range data structure
TY_FRAME_DATA
TY_GYRO_BIAS
TY_GYRO_MISALIGNMENT
TY_GYRO_SCALE
TY IMAGE DATA 24

4 Class Index

TY_IMU_DATA	24
TY_INT_RANGE	25
TY_INTERFACE_INFO	25
TY_ISP_FEATURE_INFO	26
TY_LASER_PARAM	26
TY_LASER_PATTERN_PARAM	27
TY_PHC_GROUP_ATTR	28
	28
	29
TY_TEMP_DATA	29
TY_TOF_FREQ	30
TY_TRIGGER_PARAM	30
	30
	31
TY_TRIGGER_TIMER_PERIOD	31
TY_VECT_3F	31
TY VERSION INFO	32

# **Chapter 3**

# File Index

# 3.1 File List

Here is a list of all documented files with brief descriptions:

TYApi.h	
TYApi.h includes camera control and data receiving interface, which supports configuration for	
image resolution, frame rate, exposure	
time, gain, working mode,etc	33
TYCoordinateMapper.h	
Coordinate Conversion API	02
TYDefs.h	
TYDefs.h includes camera control and data receiving data definitions which supports configura-	
tion for image resolution, frame rate, exposure	
time, gain, working mode,etc	80
TYImageProc.h	31
Tylsp.h	35
TYVer.h	??

6 File Index

# **Chapter 4**

# **Class Documentation**

# 4.1 DepthEnhenceParameters Struct Reference

#### **Public Attributes**

- float sigma\_s
   filter param on space
- float sigma\_r

filter param on range

· int outlier\_win\_sz

outlier filter windows ize

• float outlier\_rate

#### 4.1.1 Detailed Description

Definition at line 54 of file TYImageProc.h.

The documentation for this struct was generated from the following file:

• TYImageProc.h

# 4.2 DepthSpeckleFilterParameters Struct Reference

#### **Public Attributes**

- int max\_speckle\_size
- int max\_speckle\_diff

#### 4.2.1 Detailed Description

Definition at line 34 of file TYImageProc.h.

The documentation for this struct was generated from the following file:

• TYImageProc.h

# 4.3 pattern\_bin\_param Struct Reference

#### **Public Attributes**

- uint32\_t offset
- uint8\_t data [512]

#### 4.3.1 Detailed Description

Definition at line 987 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

### 4.4 pattern\_gray\_param Struct Reference

#### **Public Attributes**

- uint32\_t phase\_num
- uint32\_t param1
- uint32\_t param2
- uint32\_t param3

#### 4.4.1 Detailed Description

Definition at line 979 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

### 4.5 pattern\_sine\_param Struct Reference

#### **Public Attributes**

- uint32\_t phase\_num
- float period

### 4.5.1 Detailed Description

Definition at line 973 of file TYDefs.h.

The documentation for this struct was generated from the following file:

# 4.6 TY\_PHC\_GROUP\_ATTR::phc\_group\_attr Struct Reference

#### **Public Attributes**

- uint8\_t type
- uint8\_t amp\_thresh
- uint16\_t ch
- uint8\_t chn\_type
- uint8\_t rsvd [27]

#### 4.6.1 Detailed Description

Definition at line 957 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.7 TY\_ACC\_BIAS Struct Reference

#include <TYDefs.h>

#### **Public Attributes**

• float data [3]

#### 4.7.1 Detailed Description

a 3x3 matrix

•	•	
BIASx	BIASy	BIASz

Definition at line 1037 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.8 TY\_ACC\_MISALIGNMENT Struct Reference

#include <TYDefs.h>

#### **Public Attributes**

• float data [3 \*3]

#### 4.8.1 Detailed Description

a 3x3 matrix |.|.|.|

1	-GAMAyz	GAMAzy
GAMAxz	1	-GAMAzx
-GAMAxy	GAMAyx	1

Definition at line 1049 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.9 TY\_ACC\_SCALE Struct Reference

#include <TYDefs.h>

#### **Public Attributes**

• float **data** [3 \*3]

#### 4.9.1 Detailed Description

a 3x3 matrix

•		
SCALEx	0	0
0	SCALEy	0
0	0	SCALEz

Definition at line 1060 of file TYDefs.h.

The documentation for this struct was generated from the following file:

# 4.10 TY\_AEC\_ROI\_PARAM Struct Reference

#### **Public Attributes**

- uint32\_t x
- uint32\_t **y**
- uint32\_t w
- uint32\_t **h**

#### 4.10.1 Detailed Description

Definition at line 937 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.11 TY\_BYTEARRAY\_ATTR Struct Reference

byte array data structure

#include <TYDefs.h>

# **Public Attributes**

• int32 t size

Bytes array size in bytes.

- int32\_t unit\_size
- int32\_t valid\_size

### 4.11.1 Detailed Description

byte array data structure

See also

TYGetByteArray

Definition at line 805 of file TYDefs.h.

#### 4.11.2 Member Data Documentation

#### 4.11.2.1 unit\_size

int32\_t TY\_BYTEARRAY\_ATTR::unit\_size

unit size in bytes for special parse

Definition at line 808 of file TYDefs.h.

#### 4.11.2.2 valid\_size

```
int32_t TY_BYTEARRAY_ATTR::valid_size
```

valid size in bytes in case has reserved member, Must be multiple of unit\_size, mem\_length = valid\_size/unit\_size

Definition at line 811 of file TYDefs.h.

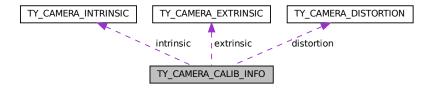
The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.12 TY\_CAMERA\_CALIB\_INFO Struct Reference

#include <TYDefs.h>

Collaboration diagram for TY\_CAMERA\_CALIB\_INFO:



#### **Public Attributes**

- int32\_t intrinsicWidth
- int32\_t intrinsicHeight
- TY\_CAMERA\_INTRINSIC intrinsic
- TY\_CAMERA\_EXTRINSIC extrinsic
- TY\_CAMERA\_DISTORTION distortion

#### 4.12.1 Detailed Description

camera 's cailbration data

See also

**TYGetStruct** 

Definition at line 880 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

### 4.13 TY\_CAMERA\_DISTORTION Struct Reference

```
#include <TYDefs.h>
```

#### **Public Attributes**

• float data [12]

Definition is compatible with opencv3.0+:k1,k2,p1,p2,k3,k4,k5,k6,s1,s2,s3,s4.

#### 4.13.1 Detailed Description

camera distortion parameters

See also

```
TYGetStruct Usage:
```

```
TY_CAMERA_DISTORTION distortion;
TYGetStruct(hDevice, some_compoent, TY_STRUCT_CAM_DISTORTION, &distortion, sizeof(distortion));
```

Definition at line 872 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

# 4.14 TY\_CAMERA\_EXTRINSIC Struct Reference

```
#include <TYDefs.h>
```

#### **Public Attributes**

• float data [4 \*4]

#### 4.14.1 Detailed Description

a 4x4 matrix

r11	r12	r13	t1
r21	r22	r23	t2
r31	r32	r33	t3
0	0	0	1

#### See also

```
TYGetStruct Usage:
TY_CAMERA_EXTRINSIC extrinsic;
TYGetStruct(hDevice, some_compoent, TY_STRUCT_EXTRINSIC, &extrinsic, sizeof(extrinsic));
```

Definition at line 860 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

# 4.15 TY\_CAMERA\_INTRINSIC Struct Reference

```
#include <TYDefs.h>
```

#### **Public Attributes**

• float data [3 \*3]

#### 4.15.1 Detailed Description

a 3x3 matrix



#### See also

```
TYGetStruct Usage:
TY_CAMERA_INTRINSIC intrinsic;
TYGetStruct(hDevice, some_compoent, TY_STRUCT_CAM_INTRINSIC, &intrinsic, sizeof(intrinsic));
```

Definition at line 842 of file TYDefs.h.

The documentation for this struct was generated from the following file:

# 4.16 TY\_CAMERA\_ROTATION Struct Reference

#include <TYDefs.h>

#### **Public Attributes**

• float data [3 \*3]

#### 4.16.1 Detailed Description

a 3x3 matrix

•	•	•
r00	r01	r02
r10	r11	r12
r20	r21	r22

#### See also

```
TYGetStruct Usage:
TY_CAMERA_ROTATION rotation;
TYGetStruct(hDevice, some_compoent, TY_STRUCT_CAM_ROTATION, &rotation, sizeof(rotation));
```

Definition at line 1218 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.17 TY\_CAMERA\_STATISTICS Struct Reference

#### **Public Attributes**

- · uint64\_t packetReceived
- uint64\_t packetLost
- uint64\_t imageOutputed
- uint64\_t imageDropped
- uint8\_t rsvd [1024]

#### 4.17.1 Detailed Description

Definition at line 1011 of file TYDefs.h.

The documentation for this struct was generated from the following file:

# 4.18 TY\_CAMERA\_TO\_IMU Struct Reference

#include <TYDefs.h>

#### **Public Attributes**

• float data [4 \*4]

#### 4.18.1 Detailed Description

a 4x4 matrix

r11	r12	r13	t1
r21	r22	r23	t2
r31	r32	r33	t3
0	0	0	1

Definition at line 1103 of file TYDefs.h.

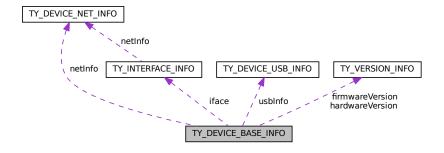
The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.19 TY\_DEVICE\_BASE\_INFO Struct Reference

#include <TYDefs.h>

Collaboration diagram for TY\_DEVICE\_BASE\_INFO:



#### **Public Attributes**

```
    TY_INTERFACE_INFO iface
    char id [32]
        device serial number
    char vendorName [32]
    char userDefinedName [32]
    char modelName [32]
        device model name
    TY_VERSION_INFO hardwareVersion
        deprecated
    TY_VERSION_INFO firmwareVersion
        deprecated
    union {
              TY_DEVICE_NET_INFO netInfo
              TY_DEVICE_USB_INFO usbInfo
        };
    char buildHash [256]
```

#### 4.19.1 Detailed Description

char configVersion [256]char reserved [256]

See also

**TYGetDeviceList** 

Definition at line 744 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.20 TY\_DEVICE\_NET\_INFO Struct Reference

```
device network information
```

```
#include <TYDefs.h>
```

#### **Public Attributes**

- char **mac** [32]
- char ip [32]
- char netmask [32]
- char gateway [32]
- char broadcast [32]
- char reserved [96]

#### 4.20.1 Detailed Description

device network information

Definition at line 716 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

### 4.21 TY DEVICE USB INFO Struct Reference

#### **Public Attributes**

- int bus
- int addr
- char reserved [248]

#### 4.21.1 Detailed Description

Definition at line 726 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

# 4.22 TY\_DI\_WORKMODE Struct Reference

#### **Public Attributes**

- TY\_E\_DI\_MODE mode
- TY\_E\_DI\_INT\_ACTION int\_act
- uint32\_t mode\_supported
- uint32\_t int\_act\_supported
- uint32\_t status
- uint32\_t reserved [3]

#### 4.22.1 Detailed Description

Definition at line 1185 of file TYDefs.h.

The documentation for this struct was generated from the following file:

# 4.23 TY\_DO\_WORKMODE Struct Reference

#### **Public Attributes**

- TY\_E\_DO\_MODE mode
- TY\_E\_VOLT\_T volt
- uint32\_t freq
- uint32 t duty
- uint32\_t mode\_supported
- uint32\_t volt\_supported
- uint32\_t reserved [3]

#### 4.23.1 Detailed Description

Definition at line 1162 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

# 4.24 TY\_ENUM\_ENTRY Struct Reference

#include <TYDefs.h>

#### **Public Attributes**

- char description [64]
- uint32\_t value
- uint32\_t reserved [3]

#### 4.24.1 Detailed Description

enum feature entry information

See also

TYGetEnumEntryInfo

Definition at line 816 of file TYDefs.h.

The documentation for this struct was generated from the following file:

TYDefs.h

#### 4.25 TY EVENT INFO Struct Reference

#### **Public Attributes**

- TY\_EVENT eventId
- · char message [124]

#### 4.25.1 Detailed Description

Definition at line 1156 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

### 4.26 TY\_FEATURE\_INFO Struct Reference

#### **Public Attributes**

· bool isValid

true if feature exists, false otherwise

TY ACCESS MODE accessMode

feature access privilege

bool writableAtRun

feature can be written while capturing

- char reserved0 [1]
- TY\_COMPONENT\_ID componentID

owner of this feature

• TY\_FEATURE\_ID featureID

feature unique id

• char name [32]

describe string

TY\_COMPONENT\_ID bindComponentID

component ID current feature bind to

TY\_FEATURE\_ID bindFeatureID

feature ID current feature bind to

- TY\_VISIBILITY\_TYPE visibility
- · char reserved [248]

#### 4.26.1 Detailed Description

Definition at line 769 of file TYDefs.h.

The documentation for this struct was generated from the following file:

# 4.27 TY\_FLOAT\_RANGE Struct Reference

float range data structure

#include <TYDefs.h>

#### **Public Attributes**

- float min
- float max
- float inc

increaing step

• float reserved [1]

#### 4.27.1 Detailed Description

float range data structure

See also

TYGetFloatRange

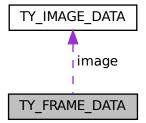
Definition at line 795 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

# 4.28 TY\_FRAME\_DATA Struct Reference

Collaboration diagram for TY\_FRAME\_DATA:



#### **Public Attributes**

void \* userBuffer

Pointer to user enqueued buffer, user should enqueue this buffer in the end of callback.

· int32\_t bufferSize

Size of userBuffer.

int32\_t validCount

Number of valid data.

• int32\_t reserved [6]

Reserved: reserved[0],laser\_val;.

• TY\_IMAGE\_DATA image [10]

Buffer data, max to 10 images per frame, each buffer data could be an image or something else.

#### 4.28.1 Detailed Description

Definition at line 1146 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.29 TY\_GYRO\_BIAS Struct Reference

#include <TYDefs.h>

#### **Public Attributes**

• float data [3]

#### 4.29.1 Detailed Description

a 3x3 matrix

•		
BIASx	BIASy	BIASz

Definition at line 1069 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

### 4.30 TY\_GYRO\_MISALIGNMENT Struct Reference

#include <TYDefs.h>

#### **Public Attributes**

• float data [3 \*3]

#### 4.30.1 Detailed Description

a 3x3 matrix

-	•	•
1	-ALPHAyz	ALPHAzy
0	1	-ALPHAzx
0	0	1

Definition at line 1080 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.31 TY\_GYRO\_SCALE Struct Reference

#include <TYDefs.h>

#### **Public Attributes**

• float data [3 \*3]

#### 4.31.1 Detailed Description

a 3x3 matrix

SCALEx	0	0
0	SCALEy	0
0	0	SCALEz

Definition at line 1091 of file TYDefs.h.

The documentation for this struct was generated from the following file:

### 4.32 TY IMAGE DATA Struct Reference

#### **Public Attributes**

uint64\_t timestamp

Timestamp in microseconds.

• int32 t imageIndex

image index used in trigger mode

· int32 t status

Status of this buffer.

TY\_COMPONENT\_ID componentID

Where current data come from.

• int32 t size

Buffer size.

void \* buffer

Pointer to data buffer.

· int32\_t width

Image width in pixels.

int32\_t height

Image height in pixels.

• TY\_PIXEL\_FORMAT pixelFormat

Pixel format, see TY\_PIXEL\_FORMAT\_LIST.

int32\_t reserved [9]

Reserved.

#### 4.32.1 Detailed Description

Definition at line 1131 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

### 4.33 TY\_IMU\_DATA Struct Reference

#### **Public Attributes**

- uint64\_t timestamp
- float acc x
- · float acc\_y
- float acc\_z
- float gyro\_x
- · float gyro\_y
- float gyro\_z
- · float temperature
- float reserved [1]

#### 4.33.1 Detailed Description

Definition at line 1020 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

### 4.34 TY\_INT\_RANGE Struct Reference

#### **Public Attributes**

- int32\_t min
- int32\_t max
- int32\_t inc

increaing step

• int32\_t reserved [1]

#### 4.34.1 Detailed Description

Definition at line 785 of file TYDefs.h.

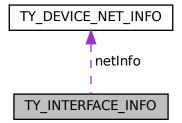
The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.35 TY\_INTERFACE\_INFO Struct Reference

#include <TYDefs.h>

Collaboration diagram for TY\_INTERFACE\_INFO:



#### **Public Attributes**

- char **name** [32]
- char id [32]
- TY\_INTERFACE\_TYPE type
- char reserved [4]
- TY\_DEVICE\_NET\_INFO netInfo

#### 4.35.1 Detailed Description

See also

**TYGetInterfaceList** 

Definition at line 734 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

### 4.36 TY\_ISP\_FEATURE\_INFO Struct Reference

#### **Public Attributes**

- TY\_ISP\_FEATURE\_ID id
- int32\_t size
- const char \* name
- const char \* value\_type
- TY\_ACCESS\_MODE mode

#### 4.36.1 Detailed Description

Definition at line 63 of file Tylsp.h.

The documentation for this struct was generated from the following file:

• Tylsp.h

# 4.37 TY\_LASER\_PARAM Struct Reference

#### **Public Attributes**

- uint32\_t idx
- uint32\_t en
- uint32\_t power

#### 4.37.1 Detailed Description

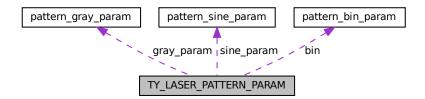
Definition at line 1121 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

# 4.38 TY\_LASER\_PATTERN\_PARAM Struct Reference

Collaboration diagram for TY\_LASER\_PATTERN\_PARAM:



#### **Public Attributes**

```
    uint32_t img_index
    uint32_t type
    union {
        uint8_t payload [512+16]
        pattern_sine_param sine_param
        pattern_gray_param gray_param
        pattern_bin_param bin
    };
```

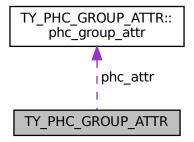
#### 4.38.1 Detailed Description

Definition at line 993 of file TYDefs.h.

The documentation for this struct was generated from the following file:

# 4.39 TY\_PHC\_GROUP\_ATTR Struct Reference

Collaboration diagram for TY\_PHC\_GROUP\_ATTR:



#### **Classes**

struct phc\_group\_attr

#### **Public Attributes**

- uint32\_t offset
- uint32\_t size
- struct TY\_PHC\_GROUP\_ATTR::phc\_group\_attr phc\_attr [16]

#### 4.39.1 Detailed Description

Definition at line 953 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.40 TY PIXEL COLOR DESC Struct Reference

### **Public Attributes**

- int16\_t x
- int16\_t y
- uint8\_t bgr\_ch1
- uint8\_t bgr\_ch2
- uint8\_t bgr\_ch3
- uint8\_t rsvd

# 4.40.1 Detailed Description

Definition at line 20 of file TYCoordinateMapper.h.

The documentation for this struct was generated from the following file:

· TYCoordinateMapper.h

# 4.41 TY\_PIXEL\_DESC Struct Reference

## **Public Attributes**

- int16\_t x
- int16\_t y
- uint16\_t depth
- uint16\_t rsvd

## 4.41.1 Detailed Description

Definition at line 12 of file TYCoordinateMapper.h.

The documentation for this struct was generated from the following file:

TYCoordinateMapper.h

# 4.42 TY\_TEMP\_DATA Struct Reference

## **Public Attributes**

- uint32\_t id
- char **name** [16]
- char **temp** [16]
- char desc [16]

# 4.42.1 Detailed Description

Definition at line 1199 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

30 Class Documentation

# 4.43 TY TOF FREQ Struct Reference

#### **Public Attributes**

- uint32\_t freq1
- · uint32\_t freq2

## 4.43.1 Detailed Description

Definition at line 1108 of file TYDefs.h.

The documentation for this struct was generated from the following file:

· TYDefs.h

# 4.44 TY\_TRIGGER\_PARAM Struct Reference

## **Public Attributes**

- TY\_TRIGGER\_MODE mode
- int8\_t fps
- int8\_t rsvd

## 4.44.1 Detailed Description

Definition at line 891 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.45 TY\_TRIGGER\_PARAM\_EX Struct Reference

#### **Public Attributes**

```
vunion {
   struct {
     int8_t fps
     int8_t duty
     int32_t laser_stream
     int32_t led_stream
     int32_t led_expo
     int32_t led_gain
   }
   struct {
     int32_t ir_gain [2]
   }
   int32_t rsvd [32]
};
```

## 4.45.1 Detailed Description

Definition at line 899 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.46 TY\_TRIGGER\_TIMER\_LIST Struct Reference

## **Public Attributes**

- uint64\_t start\_time\_us
- uint32\_t offset\_us\_count
- uint32\_t offset\_us\_list [50]

## 4.46.1 Detailed Description

Definition at line 922 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.47 TY\_TRIGGER\_TIMER\_PERIOD Struct Reference

#### **Public Attributes**

- uint64\_t start\_time\_us
- uint32\_t trigger\_count
- uint32\_t period\_us

## 4.47.1 Detailed Description

Definition at line 930 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.48 TY\_VECT\_3F Struct Reference

## **Public Attributes**

- · float x
- float y
- float z

32 Class Documentation

# 4.48.1 Detailed Description

Definition at line 823 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

# 4.49 TY\_VERSION\_INFO Struct Reference

# **Public Attributes**

- int32\_t major
- int32\_t minor
- int32\_t patch
- int32\_t reserved

# 4.49.1 Detailed Description

Definition at line 707 of file TYDefs.h.

The documentation for this struct was generated from the following file:

• TYDefs.h

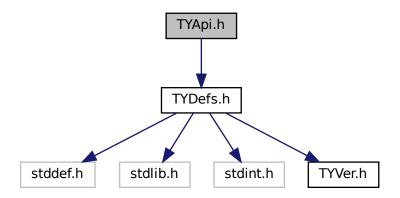
# **Chapter 5**

# **File Documentation**

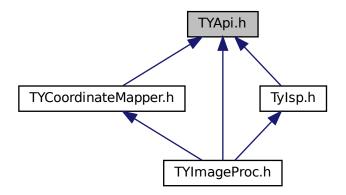
# 5.1 TYApi.h File Reference

TYApi.h includes camera control and data receiving interface, which supports configuration for image resolution, frame rate, exposure time, gain, working mode,etc.

#include "TYDefs.h"
Include dependency graph for TYApi.h:



This graph shows which files directly or indirectly include this file:



## **Typedefs**

- typedef void(\* TY\_EVENT\_CALLBACK) (TY\_EVENT\_INFO \*, void \*userdata)
- typedef void(\* TY\_IMU\_CALLBACK) (TY\_IMU\_DATA \*, void \*userdata)

#### **Functions**

- TY\_CAPI \_TYInitLib (void)
- TY\_CAPI TYLibVersion (TY\_VERSION\_INFO \*version)

Get current library version.

• TY\_EXTC const TY\_EXPORT char \*TY\_STDC TYErrorString (TY\_STATUS errorID)

Get error information.

• TY\_CAPI TYDeinitLib (void)

Deinit this library.

• TY\_CAPI TYSetLogLevel (TY\_LOG\_LEVEL lvl)

Set log level.

• TY\_CAPI TYSetLogPrefix (const char \*prefix)

set log prefix

• TY\_CAPI TYAppendLogToFile (const char \*filePath, TY\_LOG\_LEVEL lvl)

Append log to specified file.

• TY\_CAPI TYRemoveLogFile (const char \*filePath)

Remove log file.

- TY\_CAPI TYAppendLogToServer (const char \*protocol, const char \*ip, uint16\_t port, TY\_LOG\_LEVEL lvl)

  Append log to Tcp/Udp server.
- TY\_CAPI TYRemoveLogServer (const char \*protocol, const char \*ip, uint16\_t port)

Remove log server.

• TY\_CAPI TYUpdateInterfaceList (void)

Update current interfaces. call before TYGetInterfaceList.

TY\_CAPI TYGetInterfaceNumber (uint32\_t \*pNumIfaces)

Get number of current interfaces.

TY\_CAPI TYGetInterfaceList (TY\_INTERFACE\_INFO \*plfaceInfos, uint32\_t bufferCount, uint32\_t \*filled ← Count)

Get interface info list.

TY CAPI TYHasInterface (const char \*ifaceID, bool \*value)

Check if has interface.

TY\_CAPI TYOpenInterface (const char \*ifaceID, TY\_INTERFACE\_HANDLE \*outHandle)

Open specified interface.

TY CAPI TYCloseInterface (TY INTERFACE HANDLE ifaceHandle)

Close interface.

TY\_CAPI TYUpdateDeviceList (TY\_INTERFACE\_HANDLE ifaceHandle)

Update current connected devices.

TY\_CAPI TYUpdateAllDeviceList (void)

Update current connected devices.

 $\bullet \ \ TY\_CAPI\ TYGetDeviceNumber\ (TY\_INTERFACE\_HANDLE\ if ace Handle,\ uint 32\_t\ *deviceNumber)$ 

Get number of current connected devices.

TY\_CAPI TYGetDeviceList (TY\_INTERFACE\_HANDLE ifaceHandle, TY\_DEVICE\_BASE\_INFO \*device ← Infos, uint32\_t bufferCount, uint32\_t \*filledDeviceCount)

Get device info list.

TY\_CAPI TYHasDevice (TY\_INTERFACE\_HANDLE ifaceHandle, const char \*deviceID, bool \*value)
 Check whether the interface has the specified device.

 TY\_CAPI TYOpenDevice (TY\_INTERFACE\_HANDLE ifaceHandle, const char \*deviceID, TY\_DEV\_HANDLE \*outDeviceHandle, TY\_FW\_ERRORCODE \*outFwErrorcode=NULL)

Open device by device ID.

• TY\_CAPI TYOpenDeviceWithIP (TY\_INTERFACE\_HANDLE ifaceHandle, const char \*IP, TY\_DEV\_HANDLE \*deviceHandle)

Open device by device IP, useful when a device is not listed.

• TY\_CAPI TYGetDeviceInterface (TY\_DEV\_HANDLE hDevice, TY\_INTERFACE\_HANDLE \*plface)

Get interface handle by device handle.

 TY\_CAPI TYForceDeviceIP (TY\_INTERFACE\_HANDLE ifaceHandle, const char \*MAC, const char \*newIP, const char \*newNetMask, const char \*newGateway)

Force a ethernet device to use new IP address, useful when device use persistent IP and cannot be found.

• TY CAPI TYCloseDevice (TY DEV HANDLE hDevice, bool reboot=false)

Close device by device handle.

Get base info of the open device.

• TY\_CAPI TYGetDeviceInfo (TY\_DEV\_HANDLE hDevice, TY\_DEVICE\_BASE\_INFO \*info)

• TY\_CAPI TYGetComponentIDs (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID \*componentIDs) Get all components IDs.

• TY\_CAPI TYGetEnabledComponents (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID \*componentIDs)

Get all enabled components IDs.

TY\_CAPI TYEnableComponents (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentIDs)
 Enable components.

TY\_CAPI TYDisableComponents (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentIDs)
 Disable components.

TY\_CAPI TYGetFrameBufferSize (TY\_DEV\_HANDLE hDevice, uint32\_t \*bufferSize)

Get total buffer size of one frame in current configuration.

• TY CAPI TYEnqueueBuffer (TY DEV HANDLE hDevice, void \*buffer, uint32 t bufferSize)

Enqueue a user allocated buffer.

TY\_CAPI TYClearBufferQueue (TY\_DEV\_HANDLE hDevice)

Clear the internal buffer queue, so that user can release all the buffer.

• TY CAPI TYStartCapture (TY DEV HANDLE hDevice)

Start capture.

TY\_CAPI TYStopCapture (TY\_DEV\_HANDLE hDevice)
 Stop capture.

• TY\_CAPI TYSendSoftTrigger (TY\_DEV\_HANDLE hDevice)

Send a software trigger to capture a frame when device works in trigger mode.

 TY\_CAPI TYRegisterEventCallback (TY\_DEV\_HANDLE hDevice, TY\_EVENT\_CALLBACK callback, void \*userdata)

Register device status callback. Register NULL to clean callback.

• TY\_CAPI TYRegisterImuCallback (TY\_DEV\_HANDLE hDevice, TY\_IMU\_CALLBACK callback, void \*userdata)

Register imu callback. Register NULL to clean callback.

TY\_CAPI TYFetchFrame (TY\_DEV\_HANDLE hDevice, TY\_FRAME\_DATA \*frame, int32\_t timeout)
 Fetch one frame.

• TY\_CAPI TYHasFeature (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, bool \*value)

Check whether a component has a specific feature.

TY\_CAPI TYGetFeatureInfo (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, TY\_FEATURE\_INFO \*featureInfo)

Get feature info.

• TY\_CAPI TYGetIntRange (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, TY\_INT\_RANGE \*intRange)

Get value range of integer feature.

TY\_CAPI TYGetInt (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, int32\_t \*value)

Get value of integer feature.

• TY\_CAPI TYSetInt (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, int32\_t value)

Set value of integer feature.

• TY\_CAPI TYGetFloatRange (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, TY\_FLOAT\_RANGE \*floatRange)

Get value range of float feature.

• TY\_CAPI TYGetFloat (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, float \*value)

Get value of float feature.

• TY\_CAPI TYSetFloat (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, float value)

Set value of float feature.

• TY\_CAPI TYGetEnumEntryCount (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY FEATURE ID featureID, uint32 t \*entryCount)

Get number of enum entries.

• TY\_CAPI TYGetEnumEntryInfo (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, TY\_ENUM\_ENTRY \*entries, uint32\_t entryCount, uint32\_t \*filledEntryCount)

Get list of enum entries.

• TY\_CAPI TYGetEnum (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, uint32\_t \*value)

Get current value of enum feature.

• TY\_CAPI TYSetEnum (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, uint32 t value)

Set value of enum feature.

• TY\_CAPI TYGetBool (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, bool \*value)

Get value of bool feature.

• TY\_CAPI TYSetBool (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, bool value)

Set value of bool feature.

• TY\_CAPI TYGetStringLength (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY FEATURE ID featureID, uint32 t \*size)

Get internal buffer size of string feature.

• TY\_CAPI TYGetString (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, char \*buffer, uint32 t bufferSize)

Get value of string feature.

• TY\_CAPI TYSetString (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, const char \*buffer)

Set value of string feature.

• TY\_CAPI TYGetStruct (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, void \*pStruct, uint32\_t structSize)

Get value of struct.

• TY\_CAPI TYSetStruct (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, void \*pStruct, uint32\_t structSize)

Set value of struct.

• TY\_CAPI TYGetByteArraySize (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY FEATURE ID featureID, uint32 t \*pSize)

Get the size of specified byte array zone.

TY\_CAPI TYGetByteArray (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, uint8 t \*pBuffer, uint32 t bufferSize)

Read byte array from device.

• TY\_CAPI TYSetByteArray (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, const uint8\_t \*pBuffer, uint32\_t bufferSize)

Write byte array to device.

 TY\_CAPI TYGetByteArrayAttr (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_ID featureID, TY\_BYTEARRAY\_ATTR \*pAttr)

Write byte array to device.

 TY\_CAPI TYGetDeviceFeatureNumber (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, uint32\_t \*size)

Get the size of device features.

• TY\_CAPI TYGetDeviceFeatureInfo (TY\_DEV\_HANDLE hDevice, TY\_COMPONENT\_ID componentID, TY\_FEATURE\_INFO \*featureInfo, uint32\_t entryCount, uint32\_t \*filledEntryCount)

Get the all features by comp id.

• TY CAPI TYGetDeviceXMLSize (TY DEV HANDLE hDevice, uint32 t \*size)

Get the Device xml size.

TY\_CAPI TYGetDeviceXML (TY\_DEV\_HANDLE hDevice, char \*xml, const uint32\_t in\_size, uint32\_t \*out
 — size)

Get the Device xml string.

## 5.1.1 Detailed Description

TYApi.h includes camera control and data receiving interface, which supports configuration for image resolution, frame rate, exposure

time, gain, working mode,etc.

#### 5.1.2 Function Documentation

# 5.1.2.1 TYAppendLogToFile()

Append log to specified file.

#### **Parameters**

in	filePath	Path to the log file.
in <i>IVI</i>		Log level.

#### Return values

TY_STATUS_OK	Succeed.				
TY_STATUS_ERROR	Failed to add file				
	Suggestions: Please check if the file path is correct and if you have permiss	ion t	o writ	e t	.0

## 5.1.2.2 TYAppendLogToServer()

Append log to Tcp/Udp server.

#### **Parameters**

in	protocol	Protocol of the server, "tcp" or "udp".
in	ip	IP address of the server.
in	port	Port of the server.
in	lvl	Log level.

TY_STATUS_OK	Succeed.
TY_STATUS_ERROR	Failed to add server
	Suggestions: Please check if the ip and port are correct
TY_STATUS_INVALID_PARAMETER	Unsupported protocol
	Suggestions: Unsupported protocol, please use tcp or udp

## 5.1.2.3 TYClearBufferQueue()

Clear the internal buffer queue, so that user can release all the buffer.

#### **Parameters**

in <i>hDevice</i>	Device handle.
-------------------	----------------

#### Return values

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYClearBufferQueue called with invalid device handle	
	Suggestions: Please check device handle	
	Like this: TYClearBufferOueue(hDevice);	ļ
	^ is invalid	ļ
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get correct han 2.Memory in stack to store handle data is corrupted	dle
	3.After getting handle, you updated device list by calli	ng TYUpdateD
TY_STATUS_BUSY	Device is capturing.	

# 5.1.2.4 TYCloseDevice()

Close device by device handle.

## **Parameters**

in	hDevice	Device handle.
in	reboot	Reboot device after close.

TV STATUS OK	Support
TY_STATUS_UK	Succeed.

# Return values

TY_STATUS_INVALID_HANDLE	TYCloseDevice called with invalid device handle	
1	Suggestions:	•
1	Please check device handle	,
	Like this:	,
·	TYCloseDevice(hDevice, reboot);	,
·	^ is invalid	,
·	The hDevice parameter you input is not recorded	,
·	Possible reasons:	,
·	1.TYOpenDevice failed to open device and get correct handle	,
ļ	2.Memory in stack to store handle data is corrupted	7
	3.After getting handle, you updated device list by calling TYUpo	dateD
TY STATUS TIMEOUT	Failed to close device	1
,		,
	Suggestions:	,
·	Possible reasons:	<i>!</i>
·	1.Network communication is abnormal, please check whether the ne	etwor
TY_STATUS_DEVICE_ERROR	Failed to close device	
	Suggestions:	,
·	Possible reasons:	,
ļ	1.Camera device is abnormal and cannot be closed normally.	
TY STATUS IDLE	Device has been closed.	

# 5.1.2.5 TYCloseInterface()

#### Close interface.

## **Parameters**

in	ifaceHandle	Interface to be closed.
	nacci ianaic	interface to be closed.

TY_STATUS_OK	Succeed.	
TY_STATUS_NOT_INITED	TYInitLib not called.	
TY_STATUS_INVALID_INTERFACE	TYCloseInterface called with invalid interface handle	
	Suggestions:  Please check interface handle Like this:  TYCloseInterface(ifaceHandle);  ^ is invalid  The ifaceHandle parameter you input is not recorded Possible reasons:  1.TYOpenInterface failed to open interface and get 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list	d

# 5.1.2.6 TYDeinitLib()

```
TY_CAPI TYDeinitLib ( void )
```

Deinit this library.

Return values

```
TY_STATUS_OK Succeed.
```

#### 5.1.2.7 TYDisableComponents()

Disable components.

#### **Parameters**

in	hDevice	Device handle.
in	componentIDs	Components to be disabled.

## Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYDisableComponents called with invalid device handle
	Suggestions:  Please check device handle Like this:  TYDisableComponents(hDevice, componentIDs);  ^ is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUpo
TY_STATUS_INVALID_PARAMETER	Invalid component IDs
	Suggestions:  Please check componentIDs parameter  Like this:  TYDisableComponents(hDevice, componentIDs);  ^ is invalid  componentIDs should be the value returned by TYGetComponentIDs  You can also view the components of the camera by obtaining the xr

Some components specified by componentIDs are invalid.

TY\_STATUS\_INVALID\_COMPONENT

## Return values

TY_STATUS_BUSY	Camera device is capturing	
	Suggestions:  Please call TYEnableComponents when the camera dev. Like this:  TYStopCapture(hDevice);  TYDisableComponents(hDevice, componentIDs);	ice is stopped

#### See also

TY\_DEVICE\_COMPONENT\_LIST

# 5.1.2.8 TYEnableComponents()

## Enable components.

#### **Parameters**

in	hDevice	Device handle.
in	componentIDs	Components to be enabled.

TY\_STATUS\_INVALID\_COMPONENT

#### Return values

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYEnableComponents called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:  TYEnableComponents(hDevice, componentIDs);  ^ is invalid	
	The hDevice parameter you input is not recorded  Possible reasons:	
	<ol> <li>TYOpenDevice failed to open device and get cor:</li> <li>Memory in stack to store handle data is corrupt</li> </ol>	
	3.After getting handle, you updated device list b	y calling TYUpo
TY_STATUS_INVALID_PARAMETER	Invalid component IDs	
	Suggestions:	
	Please check componentIDs parameter	
	Like this:	
	TYEnableComponents(hDevice, componentIDs);	
	componentIDs should be the value returned by TYGeto You can also view the components of the camera by o	-

Some components specified by componentIDs are invalid.

TY_STATUS_BUSY	Camera device is capturing	
	Suggestions:  Please call TYEnableComponents when the camera dev.  Like this:  TYStopCapture(hDevice);  TYEnableComponents(hDevice, componentIDs);	ice is stopped

#### See also

TY\_DEVICE\_COMPONENT\_LIST

# 5.1.2.9 TYEnqueueBuffer()

Enqueue a user allocated buffer.

### **Parameters**

in	hDevice	evice Device handle.	
in	buffer	Buffer to be enqueued.	
in	bufferSize	Size of the input buffer.	

## Return values

		1
TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYEnqueueBuffer called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYEnqueueBuffer(hDevice, buffer, bufferSize);	
	^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get correct	handle
	2.Memory in stack to store handle data is corrupted	
	3.After getting handle, you updated device list by ca	lling TYUpdateD
TY_STATUS_NULL_POINTER	TYEnqueueBuffer called with NULL pointer	
	Suggestions:	
	Please check your code	
	Like this:	

TYEnqueueBuffer(hDevice, buffer, bufferSize);

^ is NULL

## Return values

TY_STATUS_WRONG_SIZE	TYEnqueueBuffer called with wrong size	
	Suggestions:	
	Please check your code	
	Like this:	
	TYEnqueueBuffer(hDevice, buffer, bufferSize);	
	^ is 0 or negative v	alue
TY_STATUS_TIMEOUT	Failed to enqueue frame buffer	
	Suggestions:	
	Possible reasons:	
	1.Network communication is abnormal, please check whet	her the networ
TY_STATUS_DEVICE_ERROR	Failed to enqueue frame buffer	
	Suggestions: Possible reasons: 1.Camera device is abnormal and cannot get the frame b	uffer size.

# 5.1.2.10 TYErrorString()

Get error information.

#### **Parameters**

in   errorID   Error id.
--------------------------

# Return values

Error	string.

## 5.1.2.11 TYFetchFrame()

```
TY_CAPI TYFetchFrame (

TY_DEV_HANDLE hDevice,

TY_FRAME_DATA * frame,

int32_t timeout )
```

Fetch one frame.

## **Parameters**

hDevice	Device handle.
frame	Frame data to be filled.
timeout	Timeout in milliseconds. <0 for infinite.
	frame

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYFetchFrame called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYFetchFrame(hDevice, pFrame, timeout);	
	^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get correct	handle
	2.Memory in stack to store handle data is corrupted	
	3.After getting handle, you updated device list by ca	lling TYUpdate
TY_STATUS_NULL_POINTER	TYFetchFrame called with NULL pointer	
	Suggestions:	
	Please check your code	
	Like this:	
	TYFetchFrame(hDevice, pFrame, timeout);	
	^ is NULL	
TY_STATUS_IDLE	Camera device is not started	
	Suggestions:	
	Please start the camera device first	
	Like this:	
	TYStartCapture(hDevice);	
	TYFetchFrame(hDevice, pFrame, timeout);	
TY STATUS WRONG MODE	Callback has been registered, this function is disabled.	
	<u> </u>	
TY_STATUS_TIMEOUT	Failed to get frame	
	Suggestions:	
	Possible reasons:	
	1.Camera device is abnormal and cannot get frame.	
	2.Network communication is abnormal, please check whe	ther the netwo
	3. Timeout, frame acquisition timeout	

## 5.1.2.12 TYForceDeviceIP()

```
TY_CAPI TYForceDeviceIP (

TY_INTERFACE_HANDLE ifaceHandle,

const char * MAC,

const char * newIP,

const char * newNetMask,

const char * newGateway )
```

Force a ethernet device to use new IP address, useful when device use persistent IP and cannot be found.

#### **Parameters**

in	ifaceHandle	Interface handle.
in	MAC	Device MAC, should be "xx:xx:xx:xx:xx:xx".
in	newIP	New IP.
in	newNetMask	New subnet mask.
in	newGateway	New gateway.

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	TYForceDeviceIP called with invalid interface handle
	Suggestions:  Please check interface handle Like this:  TYForceDeviceIP(ifaceHandle, MAC, newIP, newNetMask, newGateway)  ^ is invalid  The ifaceHandle parameter you input is not recorded Possible reasons:  1.TYOpenInterface failed to open interface and get correct handl 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling TY
TY_STATUS_WRONG_TYPE	TYForceDeviceIP called with invalid interface type
	Suggestions:  Please check interface type Usually you can get interface information by calling TYGetInterface You can use TYIsNetworkInterface to check the interface type Only network interfaces can call TYForceDeviceIP Like this:  TY_INTERFACE_INFO info; uint32_t num; TYGetInterfaceList(&info, 1, #); if(TYIsNetworkInterface(info[0].type)) {     TY_INTERFACE_HANDLE hIface;     TYOpenInterface(info[0].id, &hIface);     TYForceDeviceIP(hIface, MAC, newIP, newNetMask, newGateway); }
TY_STATUS_NULL_POINTER	TYForceDeviceIP called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYForceDeviceIP(ifaceHandle, MAC, newIP, newNetMask, newGateway)  ^ or ^ or ^ or ^ or ^ is NULL
TY_STATUS_INVALID_PARAMETER	Invalid MAC address:
	Suggestions: Please check MAC parameter Like this: TYForceDeviceIP(ifaceHandle, MAC, newIP, newNetMask, newGateway) ^ is invalid MAC address should be six bytes of hexadecimal separated by colons For example: 00:11:22:aa:bb:cc
TY_STATUS_TIMEOUT	Failed to force set IP
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne 2.There is no camera with a matching target MAC address in the r
TY_STATUS_DEVICE_ERROR	Failed to force set IP
	Suggestions: Possible reasons: 1.New IP, NetMask, Gateway are incorrect, camera device refuses

# 5.1.2.13 TYGetBool()

```
TY_CAPI TYGetBool (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

bool * value )
```

#### Get value of bool feature.

## **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	value	Bool value.

### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetBool called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:
	TYGetBool(hDevice, componentID, featureID, pValue);  ^ is invalid
	The hDevice parameter you input is not recorded
	Possible reasons:
	1.TYOpenDevice failed to open device and get correct handle
	2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUp
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:
	Please check componentID parameter
	Like this:
	TYGetBool(hDevice, componentID, featureID, pValue);
	^ is invalid componentID should be the value returned by TYGetComponentIDs
	You can also view the components of the camera by obtaining the x
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:
	Please check featureID parameter
	Like this:
	TYGetBool(hDevice, componentID, featureID, pValue);  ^ is invalid
	You entered an invalid featureID parameter
	You can get a list of features of the camera device through TYGet
	You can also view the features of the camera device by obtaining
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:
	Please check the feature type
	Like this:
	TYGetBool(hDevice, componentID, featureID, pValue);  ^ type mismatch
	The feature type you entered does not match. You can use TVFeature

The feature type you entered does not match. You can use TYFeature

## Return values

	TY_STATUS_NULL_POINTER	TYGetBool called with NULL pointer	1
		Suggestions:  Please check your code  Like this:  TYGetBool(hDevice, componentID, featureID, pValue  ^ is N	
	TY_STATUS_TIMEOUT	Failed to get bool feature	ı
		Suggestions: Possible reasons: 1.Network communication is abnormal, please check	whether the no
Ī	TY_STATUS_DEVICE_ERROR	Failed to get bool feature	1
		Suggestions:  Possible reasons:  1.The feature of the camera device is not availab 2.Camera device is abnormal and cannot get bool f	_

# 5.1.2.14 TYGetByteArray()

Read byte array from device.

#### **Parameters**

in	hDevice	Device handle.
in <i>componentID</i>		Component ID.
in	featureID	Feature ID.
out	pBuffer	Byte buffer.
in	bufferSize	Size of buffer.

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYGetByteArray called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYGetByteArray(hDevice, componentID, featureID, k ^ is invalid	uffer, bufferS
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get corn	rect handle
	2.Memory in stack to store handle data is corrupt	
	3.After getting handle, you updated device list	

TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:  Please check componentID parameter  Like this:  TYGetByteArray(hDevice, componentID, featureID, buffer, bufferS:  ^ is invalid  componentID should be the value returned by TYGetComponentIDs  You can also view the components of the camera by obtaining the xm
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:  Please check featureID parameter  Like this:  TYGetByteArray(hDevice, componentID, featureID, buffer, bufferS:  ^ is invalid  You entered an invalid featureID parameter  You can get a list of features of the camera device through TYGetI  You can also view the features of the camera device by obtaining the camera device of the camera device by obtaining the camera device of the camera device
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type  Like this:  TYGetByteArray(hDevice, componentID, featureID, buffer, bufferS:  ^ type mismatch  The feature type you entered does not match. You can use TYFeature
TY_STATUS_NULL_POINTER	TYGetByteArray called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYGetByteArray(hDevice, componentID, featureID, pBuffer, buffers)  ^ is NULL
TY_STATUS_WRONG_SIZE	Array size mismatch
	Suggestions:  Please check the array size  Like this:  TYGetByteArray(hDevice, componentID, featureID, buffer, bufferS:  ^ is interpretable to the component of the array size you entered does not match
TY_STATUS_TIMEOUT	Failed to get byte array feature
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get byte array feature
	Suggestions: Possible reasons: 1.The feature of the camera device is not available or not imple 2.Camera device is abnormal and cannot get byte array feature

# 5.1.2.15 TYGetByteArrayAttr()

```
TY_COMPONENT_ID componentID,
TY_FEATURE_ID featureID,
TY_BYTEARRAY_ATTR * pAttr )
```

Write byte array to device.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	pAttr	Byte array attribute to be filled.

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetByteArrayAttr called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:
	TYGetByteArrayAttr(hDevice, componentID, featureID, pAttr);  ^ is invalid
	The hDevice parameter you input is not recorded
	Possible reasons: 1.TYOpenDevice failed to open device and get correct handle
	2. Memory in stack to store handle data is corrupted
	3. After getting handle, you updated device list by calling TYUr
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:
	Please check componentID parameter
	Like this:
	TYGetByteArrayAttr(hDevice, componentID, feature D, pAttr);  ^ is invalid
	componentID should be the value returned by TYGetComponentIDs
	You can also view the components of the camera by obtaining the x
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:
	Please check featureID parameter
	Like this:
	TYGetByteArrayAttr(hDevice, componentID, feature D, pAttr); ^ is invalid
	You entered an invalid featureID parameter
	You can get a list of features of the camera device through TYGet You can also view the features of the camera device by obtaining
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:
	Please check the feature type
	Like this:
	TYGetByteArrayAttr(hDevice, componentID, feature D, pAttr);  ^ type mismatch
	The feature type you entered does not match. You can use TYFeatur

TY_STATUS_NULL_POINTER	TYGetByteArrayAttr called with NULL pointer	
	Suggestions: Please check your code Like this: TYGetByteArrayAttr(hDevice, componentID, feature)	ID, pAttr); ^ is NULL

# 5.1.2.16 TYGetByteArraySize()

```
TY_CAPI TYGetByteArraySize (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

uint32_t * pSize )
```

Get the size of specified byte array zone.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	pSize	Size of specified byte array zone.

## Return values

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYGetByteArraySize called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYGetByteArraySize(hDevice, componentID, feature D, pSize); ^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get correct handl	
	2.Memory in stack to store handle data is corrupted	
	3.After getting handle, you updated device list by calling TYUp	
TY_STATUS_INVALID_COMPONENT	Invalid component ID	
	Suggestions:	
	Please check componentID parameter	
	Like this:	
	TYGetByteArraySize(hDevice, componentID, feature D, pSize); ^ is invalid	
	componentID should be the value returned by TYGetComponentIDs	

You can also view the components of the camera by obtaining the x

## Return values

TY_STATUS_INVALID_FEATURE	Invalid feature ID	
	Suggestions:	
	Please check featureID parameter	
	Like this:	
	TYGetByteArraySize(hDevice, componentID, feature:	
	You entered an invalid featureID parameter	
	You can get a list of features of the camera device	through TYGet:
	You can also view the features of the camera device	by obtaining
TY_STATUS_WRONG_TYPE	Feature type mismatch	
	Suggestions:	
	Please check the feature type	
	Like this:	
	TYGetByteArraySize(hDevice, componentID, feature:	D, pSize);
	1	nismatch
	The feature type you entered does not match. You ca	n use TYFeatur
TY_STATUS_NULL_POINTER	TYGetByteArraySize called with NULL pointer	
	Suggestions:	
	Please check your code	
	Like this:	
	TYGetByteArraySize(hDevice, componentID, feature:	D, pSize); ^ is NULL

# 5.1.2.17 TYGetComponentIDs()

# Get all components IDs.

## **Parameters**

in	hDevice	Device handle.
out	componentIDs	All component IDs this device has. (bit flag).

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYGetComponentIDs called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYGetComponentIDs(hDevice, outComponentIDs);	
	^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get correct	handle
	2.Memory in stack to store handle data is corrupted	
	3.After getting handle, you updated device list by ca	lling TYUpd

TY_STATUS_NULL_POINTER	TYGetComponentIDs called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYGetComponentIDs(hDevice, outComponentIDs);  ^ is NULL

#### See also

TY\_DEVICE\_COMPONENT\_LIST

# 5.1.2.18 TYGetDeviceFeatureInfo()

```
TY_CAPI TYGetDeviceFeatureInfo (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_INFO * featureInfo,

uint32_t entryCount,

uint32_t * filledEntryCount)
```

## Get the all features by comp id.

## **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
out	featureInfo	Output feature info.
in	entryCount	Array size of input parameter "featureInfo".
out	filledEntryCount	Number of filled featureInfo.

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYGetDeviceFeatureInfo called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYGetDeviceFeatureInfo(hDevice, componentID, feat	ureInfo, entry(
	^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get corn	rect handle
	2.Memory in stack to store handle data is corrupt	ted
	3.After getting handle, you updated device list h	

## Return values

TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:  Please check componentID parameter  Like this:  TYGetDeviceFeatureInfo(hDevice, componentID, featureInfo, entry(  ^ is invalid  componentID should be the value returned by TYGetComponentIDs  You can also view the components of the camera by obtaining the xr
TY_STATUS_NULL_POINTER	TYGetDeviceFeatureInfo called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYGetDeviceFeatureInfo(hDevice, componentID, featureInfo, entry)  or
TY_STATUS_TIMEOUT	Failed to get feature info
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get feature info
	Suggestions: Possible reasons: 1.The feature of the camera device is not available or not imple 2.Camera device is abnormal and cannot get feature info

# 5.1.2.19 TYGetDeviceFeatureNumber()

Get the size of device features.

## **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
out	size	Size of all feature cnt.

TY_STATUS_OK	Succeed.

TY_STATUS_INVALID_HANDLE	TYGetDeviceFeatureNumber called with invalid device handle	
	Suggestions:  Please check device handle Like this:  TYGetDeviceFeatureNumber(hDevice, componentID, si  ^ is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get corr 2.Memory in stack to store handle data is corrupt 3.After getting handle, you updated device list b	ect handle ed
TY_STATUS_INVALID_COMPONENT	Invalid component ID	
	Suggestions:  Please check componentID parameter  Like this:  TYGetDeviceFeatureNumber(hDevice, componentID, si  ^ is invalid  componentID should be the value returned by TYGetCo  You can also view the components of the camera by	mponentIDs
TY_STATUS_NULL_POINTER	TYGetDeviceFeatureNumber called with NULL pointer	
	Suggestions:  Please check your code  Like this:  TYGetDeviceFeatureNumber(hDevice, componentID, si	ze); is NULL
TY_STATUS_TIMEOUT	Failed to get feature number	
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check	whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get feature number	
	Suggestions: Possible reasons: 1.The feature of the camera device is not availab 2.Camera device is abnormal and cannot get featur	÷

# 5.1.2.20 TYGetDeviceInfo()

Get base info of the open device.

## **Parameters**

	in	hDevice	Device handle.
ſ	out	info	Base info out.

## Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetDeviceInfo called with invalid device handle
	Suggestions:  Please check device handle Like this:  TYGetDeviceInfo(hDevice, info);  ^ is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUpdateD
TY_STATUS_NULL_POINTER	TYGetDeviceInfo called with NULL pointer
	Suggestions:  Please check your code Like this:  TYGetDeviceInfo(hDevice, info);  ^ is NULL

# 5.1.2.21 TYGetDeviceInterface()

Get interface handle by device handle.

# **Parameters**

in	hDevice	Device handle.
out	plface	Interface handle.

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetDeviceInterface called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:
	TYGetDeviceInterface(hDevice, pIface);
	^ is invalid
	The hDevice parameter you input is not recorded
	Possible reasons:
	1.TYOpenDevice failed to open device and get correct handle
	2.Memory in stack to store handle data is corrupted
	3.After getting handle, you updated device list by calling TYUpo

TY_STATUS_NULL_POINTER	TYGetDeviceInterface called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYGetDeviceInterface(hDevice, pIface);  ^ is NULL

# 5.1.2.22 TYGetDeviceList()

```
TY_CAPI TYGetDeviceList (

TY_INTERFACE_HANDLE ifaceHandle,

TY_DEVICE_BASE_INFO * deviceInfos,

uint32_t bufferCount,

uint32_t * filledDeviceCount )
```

## Get device info list.

#### **Parameters**

in	ifaceHandle	Interface handle.
out	deviceInfos	Device info array to be filled.
in	bufferCount	Array size of deviceInfos.
out	filledDeviceCount	Number of filled TY_DEVICE_BASE_INFO.

TY_STATUS_OK	Succeed.	
TY_STATUS_NOT_INITED	TYInitLib not called.	
TY_STATUS_INVALID_INTERFACE	TYGetDeviceList called with invalid interface handle	
	Suggestions:  Please check interface handle Like this:  TYGetDeviceList(ifaceHandle, pDeviceInfos, bufferCount, pFille  ^ is invalid  The ifaceHandle parameter you input is not recorded Possible reasons:  1.TYOpenInterface failed to open interface and get correct han 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling	ndle
TV OTATUO NUUL BOUNTER		
TY_STATUS_NULL_POINTER	TYGetDeviceList called with NULL pointer	
	Suggestions:  Please check your code  Like this:  TYGetDeviceList(ifaceHandle, pDeviceInfos, bufferCount, pFille  ^ is NULL or ^ is 0 or ^ is 1	

# 5.1.2.23 TYGetDeviceNumber()

```
TY_CAPI TYGetDeviceNumber (
            TY_INTERFACE_HANDLE ifaceHandle,
            uint32_t * deviceNumber)
```

Get number of current connected devices.

#### **Parameters**

in	ifaceHandle	Interface handle.
out	deviceNumber	Number of connected devices.

#### Return values

TY STATUS OK	Succeed.	
TY_STATUS_NOT_INITED	TYInitLib not called.	
TY_STATUS_INVALID_INTERFACE	TYGetDeviceNumber called with invalid interface handle	
	Suggestions: Please check interface handle Like this: TYGetDeviceNumber(ifaceHandle, pDeviceNumber);  ^ is invalid The ifaceHandle parameter you input is not recorded Possible reasons: 1.TYOpenInterface failed to open interface and get 2.Memory in stack to store handle data is corrupte 3.After getting handle, you updated interface list	d
TY STATUS NULL POINTER	TYGetDeviceNumber called with NULL pointer	
	Suggestions:  Please check your code  Like this:  TYGetDeviceNumber(ifaceHandle, deviceNumber);  ^ is NULL	

## 5.1.2.24 TYGetDeviceXML()

```
TY_CAPI TYGetDeviceXML (
            TY_DEV_HANDLE hDevice,
            char * xml,
            const uint32_t in_size,
            uint32_t * out_size )
```

Get the Device xml string.

#### **Parameters**

in	hDevice	Device handle.
in	xml	The buffer to store xml
in	in_size	The size buffer
out	out_size	The actual size write in buffer

TY_STATUS_OK	Succeed.	
TY_STATUS_NOT_INITED	Not call TYInitLib	
TY_STATUS_WRONG_SIZE	XML buffer size is not enough	
	Suggestions:	
	XML buffer size is not enough	
	Like this:	
	TYGetDeviceXML(hDevice, xml, in_size, out_size);	
	^ is invalid	
	XML buffer size is not enough, please use TYGetDeviceXM	LSize to get th
TY_STATUS_INVALID_HANDLE	TYGetDeviceXML called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYGetDeviceXML(hDevice, xml, in_size, out_size);	
	^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	h
	1.TYOpenDevice failed to open device and get correct 2.Memory in stack to store handle data is corrupted	hanqie
	3. After getting handle, you updated device list by cal	lling TYUpdateD
		11119 1107
TY_STATUS_NULL_POINTER	TYGetDeviceXML called with NULL pointer	
	Suggestions:	İ
	Please check your code	  -
	Like this:	  -
	TYGetDeviceXML(hDevice, xml, in_size, out_size);	
		İ

# 5.1.2.25 TYGetDeviceXMLSize()

Get the Device xml size.

#### **Parameters**

in	hDevice	Device handle.
out	size	The size of device xml string

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	Not call TYInitLib

## Return values

TY_STATUS_INVALID_HANDLE	TYGetDeviceXMLSize called with invalid device handle	
	Suggestions:  Please check device handle Like this:  TYGetDeviceXMLSize(hDevice, size);  ^ is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get correct 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by ca	
TY_STATUS_NULL_POINTER	TYGetDeviceXMLSize called with NULL pointer	
	Suggestions:  Please check your code  Like this:  TYGetDeviceXMLSize(hDevice, size);  ^ is NULL	

# 5.1.2.26 TYGetEnabledComponents()

Get all enabled components IDs.

#### **Parameters**

in	hDevice	Device handle.
out	componentIDs	Enabled component IDs.(bit flag)

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TATUS_INVALID_HANDLE  TYGetEnabledComponents called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	<pre>TYGetEnabledComponents(hDevice, componentIDs);</pre>	
	^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get correct	handle
	2.Memory in stack to store handle data is corrupted	
	3.After getting handle, you updated device list by ca	lling TYUpdateI
TY STATUS NULL POINTER	componentIDs is NHH I	

#### See also

TY\_DEVICE\_COMPONENT\_LIST

# 5.1.2.27 TYGetEnum()

Get current value of enum feature.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	value	Enum value.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetEnum called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:
	TYGetEnum(hDevice, componentID, featureID, pValue);  ^ is invalid
	The hDevice parameter you input is not recorded Possible reasons:
	1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted
	3.After getting handle, you updated device list by calling TYUp
TY_STATUS_INVALID_COMPONENT	Invalid component ID: d
	Suggestions:
	Please check componentID parameter
	Like this:
	TYGetEnum(hDevice, componentID, featureID, pValue);  ^ is invalid
	componentID should be the value returned by TYGetComponentIDs
	You can also view the components of the camera by obtaining the x
TY_STATUS_INVALID_FEATURE	Invalid feature ID: d
	Suggestions:
	Please check featureID parameter
	Like this:
	TYGetEnum(hDevice, componentID, featureID, pValue);

You entered an invalid featureID parameter

^ is invalid

You can get a list of features of the camera device through TYGet. You can also view the features of the camera device by obtaining

## Return values

TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:
	Please check the feature type
	Like this:  TYGetEnum(hDevice, componentID, featureID, pValue);
	^ type mismatch
	The feature type you entered does not match. You can use TYFeature
TY_STATUS_NULL_POINTER	TYGetEnum called with NULL pointer
	Suggestions:
	Please check your code
	Like this:
	TYGetEnum(hDevice, componentID, featureID, pValue);  ^ is NULL
TY_STATUS_TIMEOUT	Failed to get enum feature
	Suggestions:
	Possible reasons:
	1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get enum feature
	Suggestions:
	Possible reasons:  1. The feature of the camera device is not available or not imple
	2. Camera device is abnormal and cannot get enum feature

# 5.1.2.28 TYGetEnumEntryCount()

```
TY_CAPI TYGetEnumEntryCount (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

uint32_t * entryCount )
```

#### Get number of enum entries.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	entryCount	Entry count.

TY_STATUS_OK	Succeed.

TY_STATUS_INVALID_HANDLE	TYGetEnumEntryCount called with invalid device handle
	Suggestions:  Please check device handle  Like this:  TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount
	^ is invalid The hDevice parameter you input is not recorded Possible reasons: 1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUpo
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:  Please check componentID parameter  Like this:  TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount
	^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the xr
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:  Please check featureID parameter  Like this:  TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount  ^ is invalid  You entered an invalid featureID parameter  You can get a list of features of the camera device through TYGetI  You can also view the features of the camera device by obtaining the state of the camera device of the camera device by obtaining the state of the camera device of the camera device of the camera device by obtaining the state of the camera device of the camera
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type Like this:  TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount  ^ type mismatch  The feature type you entered does not match. You can use TYFeature
TY_STATUS_NULL_POINTER	TYGetEnumEntryCount called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYCOTERUMENT TO COURT (brown to component ID feature ID pentry Court
	TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount

# 5.1.2.29 TYGetEnumEntryInfo()

```
TY_CAPI TYGetEnumEntryInfo (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

TY_ENUM_ENTRY * entries,

uint32_t entryCount,

uint32_t * filledEntryCount )
```

Get list of enum entries.

# **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	entries	Output entries.
in	entryCount	Array size of input parameter "entries".
out	filledEntryCount	Number of filled entries.

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetEnumEntryInfo called with invalid device handle
	Suggestions:  Please check device handle Like this:  TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, ent  is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUpo
TY_STATUS_INVALID_COMPONENT	Invalid component ID: d
	Suggestions:  Please check componentID parameter  Like this:  TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, ent  ^ is invalid  componentID should be the value returned by TYGetComponentIDs  You can also view the components of the camera by obtaining the xr
TY_STATUS_INVALID_FEATURE	Invalid feature ID: d
	Suggestions:  Please check featureID parameter Like this:  TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, ent  ^ is invalid  You entered an invalid featureID parameter  You can get a list of features of the camera device through TYGetI  You can also view the features of the camera device by obtaining t
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type Like this:  TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, ent  ^ type mismatch  The feature type you entered does not match. You can use TYFeature
TY_STATUS_NULL_POINTER	TYGetEnumEntryInfo called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYGetEnumEntryInfo(hDevice, componentID, featureID, pEnumDescrip

# 5.1.2.30 TYGetFeatureInfo()

```
TY_CAPI TYGetFeatureInfo (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

TY_FEATURE_INFO * featureInfo )
```

#### Get feature info.

## **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	featureInfo	Feature info.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetFeatureInfo called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:
	TYGetFeatureInfo(hDevice, componentID, featureID, pFeatureInfo
	The hDevice parameter you input is not recorded  Possible reasons:
	1.TYOpenDevice failed to open device and get correct handle
	2.Memory in stack to store handle data is corrupted
	3.After getting handle, you updated device list by calling TY
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:
	Please check componentID parameter
	Like this:
	TYGetFeatureInfo(hDevice, componentID, featureID, pFeatureInfo
	componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the
	Tou can also view the components of the camera by obtaining the
TY_STATUS_INVALID_COMPONENT	Invalid feature ID
	Suggestions:
	Please check featureID parameter
	Like this:
	TYGetFeatureInfo(hDevice, componentID, featureID, pFeatureInfo
	You entered an invalid featureID parameter
	You can get a list of features of the camera device through TYG
	You can also view the features of the camera device by obtaining
TY_STATUS_NULL_POINTER	TYGetFeatureInfo called with NULL pointer
	Suggestions:
	Please check your code
	Like this:
	myggi miggi the control of the contr

TYGetFeatureInfo(hDevice, componentID, featureID, pFeatureInfo)

^ is NULL

# 5.1.2.31 TYGetFloat()

```
TY_CAPI TYGetFloat (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

float * value )
```

#### Get value of float feature.

## **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	value	Float value.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetFloat called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:
	TYGetFloat(hDevice, componentID, featureID, pValue); ^ is invalid
	The hDevice parameter you input is not recorded
	Possible reasons:
	1.TYOpenDevice failed to open device and get correct handle
	2. Memory in stack to store handle data is corrupted
	3.After getting handle, you updated device list by calling TYUp
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:
	Please check componentID parameter
	Like this:
	TYGetFloat(hDevice, componentID, featureID, pValue);
	componentID should be the value returned by TYGetComponentIDs
	You can also view the components of the camera by obtaining the x
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:
	Please check featureID parameter
	Like this:
	TYGetFloat(hDevice, componentID, featureID, pValue);
	You entered an invalid featureID parameter
	You can get a list of features of the camera device through TYGet
	You can also view the features of the camera device by obtaining
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:
	Please check the feature type
	Like this:
	TYGetFloat(hDevice, componentID, featureID, pValue);

^ type mismatch

The feature type you entered does not match. You can use TYFeature

	TY_STATUS_NULL_POINTER	TYGetFloat called with NULL pointer		
		Suggestions:  Please check your code  Like this:  TYGetFloat(hDevice, componentID, featureID, pValu  ^ is	ue); NULL	
Ī	TY_STATUS_TIMEOUT	Failed to get float feature		
		Suggestions: Possible reasons: 1.Network communication is abnormal, please check	k whether the	ne
Ī	TY_STATUS_DEVICE_ERROR	Failed to get float feature		
		Suggestions:  Possible reasons:  1.The feature of the camera device is not available 2.Camera device is abnormal and cannot get float		pl

# 5.1.2.32 TYGetFloatRange()

```
TY_CAPI TYGetFloatRange (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

TY_FLOAT_RANGE * floatRange )
```

Get value range of float feature.

### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	floatRange	Float range to be filled.

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYGetFloatRange called with invalid device handle	
	Suggestions: Please check device handle Like this: TYGetFloatRange(hDevice, componentID, featureID, ^ is invalid The hDevice parameter you input is not recorded Possible reasons: 1.TYOpenDevice failed to open device and get corn 2.Memory in stack to store handle data is corrupt 3.After getting handle, you updated device list h	rect handle

## Return values

TY_STATUS_INVALID_COMPONENT	Invalid component ID	
	Suggestions:  Please check componentID parameter  Like this:  TYGetFloatRange(hDevice, componentID, featureID,  ^ is invalid  componentID should be the value returned by TYGetCo	
	You can also view the components of the camera by a	±
TY_STATUS_INVALID_FEATURE	Invalid feature ID	
	Suggestions:  Please check featureID parameter Like this:  TYGetFloatRange(hDevice, componentID, featureID,  ^ is inval: You entered an invalid featureID parameter You can get a list of features of the camera device You can also view the features of the camera device	d through TYGetI
TY_STATUS_WRONG_TYPE	Feature type mismatch	
	Suggestions:  Please check the feature type Like this:  TYGetFloatRange(hDevice, componentID, featureID,  ^ type miss  The feature type you entered does not match. You ca	natch
TY_STATUS_NULL_POINTER	TYGetFloatRange called with NULL pointer	
	Suggestions: Please check your code Like this: TYGetFloatRange(hDevice, componentID, featureID,	pFloatRange); ^ is NULL

# 5.1.2.33 TYGetFrameBufferSize()

Get total buffer size of one frame in current configuration.

## **Parameters**

in	hDevice	Device handle.
out	bufferSize	Buffer size per frame.

TY_STATUS_OK	Succeed.

TY_STATUS_INVALID_HANDLE	TYGetFrameBufferSize called with invalid device handle
	Suggestions:  Please check device handle Like this:  TYGetFrameBufferSize(hDevice, outSize);  ^ is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUpdateD
TY_STATUS_NULL_POINTER	TYGetFrameBufferSize called with NULL pointer
	Suggestions: Please check your code Like this: TYGetFrameBufferSize(hDevice, outSize);  ^ is NULL
TY_STATUS_TIMEOUT	Failed to get frame buffer size
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network
TY_STATUS_DEVICE_ERROR	Failed to get frame buffer size
	Suggestions: Possible reasons: 1.Camera device is abnormal and cannot get the frame buffer size.

# 5.1.2.34 TYGetInt()

Get value of integer feature.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	value	Integer value.

TY_STATUS_OK	Succeed.

TY_STATUS_INVALID_HANDLE	TYGetInt called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:  TYGetInt(hDevice, componentID, featureID, pValue);  ^ is invalid
	The hDevice parameter you input is not recorded Possible reasons:
	1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUpo
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:  Please check componentID parameter  Like this:  TYGetInt(hDevice, componentID, featureID, pValue);
	^ is invalid componentID should be the value returned by TYGetComponentIDs
	You can also view the components of the camera by obtaining the xr
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions: Please check featureID parameter Like this: TYGetInt(hDevice, componentID, featureID, pValue);
	^ is invalid You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetI You can also view the features of the camera device by obtaining t
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type Like this:  TYGetInt(hDevice, componentID, featureID, pValue);  ^ type mismatch The feature type you entered does not match. You can use TYFeature
TY_STATUS_NULL_POINTER	TYGetInt called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYGetInt(hDevice, componentID, featureID, pValue);  ^ is NULL
TY_STATUS_TIMEOUT	Failed to get int feature
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get int feature
	Suggestions: Possible reasons: 1.The feature of the camera device is not available or not imple 2.Camera device is abnormal and cannot get int feature

# 5.1.2.35 TYGetInterfaceList()

#### Get interface info list.

#### **Parameters**

out	plfaceInfos	Array of interface infos to be filled.
in	bufferCount	Array size of interface infos.
out	filledCount	Number of filled TY_INTERFACE_INFO.

## Return values

TY_STATUS_OK	Succeed.	
TY_STATUS_NOT_INITED	TYInitLib not called.	
TY_STATUS_NULL_POINTER	TYGetInterfaceList called with NULL pointer	
	Suggestions:  Please check your code  Like this:  TYGetInterfaceList(pIfaceInfos, bufferCount, filledCount  ^ or ^ is NULL	:);

# 5.1.2.36 TYGetInterfaceNumber()

```
TY_CAPI TYGetInterfaceNumber ( \mbox{uint32\_t} \ * \ p\mbox{\it NumIfaces} \ )
```

#### Get number of current interfaces.

### Parameters

out	pNumIfaces	Number of interfaces.
040	priamacco	Trainibol of intollaces.

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	TYGetInterfaceNumber called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYGetInterfaceNumber(pNumIfaces);  ^ is NULL

# 5.1.2.37 TYGetIntRange()

```
TY_CAPI TYGetIntRange (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

TY_INT_RANGE * intRange )
```

Get value range of integer feature.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	intRange	Integer range to be filled.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetIntRange called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:
	TYGetIntRange(hDevice, componentID, featureID, pintRange); ^ is invalid
	The hDevice parameter you input is not recorded Possible reasons:
	1.TYOpenDevice failed to open device and get correct handle
	2.Memory in stack to store handle data is corrupted
	3.After getting handle, you updated device list by calling TYU
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:
	Please check componentID parameter
	Like this:
	TYGetIntRange(hDevice, componentID, featureID, pintRange);  ^ is invalid
	componentID should be the value returned by TYGetComponentIDs
	You can also view the components of the camera by obtaining the
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:
	Please check featureID parameter
	Like this:
	TYGetIntRange(hDevice, componentID, featureID, pIntRange);  ^ is invalid
	You entered an invalid featureID parameter
	You can get a list of features of the camera device through TYG

You can also view the features of the camera device by obtaining

TY_STATUS_WRONG_TYPE	Feature type mismatch	
	Suggestions:	
	Please check the feature type	
	Like this:	
	TYGetIntRange(hDevice, componentID, featureID, p	ntRange);
	^ type mismat	ch
	The feature type you entered does not match. You ca	n use TYFeature
TY STATUS NULL POINTER	TYGetIntRange called with NULL pointer	
	·	
	Suggestions:	
	Please check your code	
	Like this:	
	TYGetIntRange(hDevice, componentID, featureID, p	ntRange);
	^	is NULL

# 5.1.2.38 TYGetString()

## Get value of string feature.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	buffer	String buffer.
in	bufferSize	Size of buffer.

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYGetString called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYGetString(hDevice, componentID, featureID, bufi ^ is invalid	fer, bufferSize)
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get corr	ect handle
	2.Memory in stack to store handle data is corrupt	ed
	3.After getting handle, you updated device list b	py calling TYUpo

## Return values

TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:  Please check componentID parameter  Like this:  TYGetString(hDevice, componentID, featureID, buffer, bufferSize)
	^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the xr
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:  Please check featureID parameter  Like this:  TYGetString(hDevice, componentID, featureID, buffer, bufferSize)  ^ is invalid
	You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetI You can also view the features of the camera device by obtaining t
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type  Like this:  TYGetString(hDevice, componentID, featureID, buffer, bufferSize)  ^ type mismatch  The feature type you entered does not match. You can use TYFeature
TY_STATUS_NULL_POINTER	TYGetString called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYGetString(hDevice, componentID, featureID, pBuffer, bufferSize  ^ is NULL
TY_STATUS_TIMEOUT	Failed to get string feature
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get string feature
	Suggestions: Possible reasons: 1.The feature of the camera device is not available or not imple 2.Camera device is abnormal and cannot get string feature

## See also

TYGetStringLength

# 5.1.2.39 TYGetStringLength()

```
TY_FEATURE_ID featureID,
uint32_t * size )
```

Get internal buffer size of string feature.

## **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	size	String length including '\0'.

## Return values

TV CTATUS OK	Outside
TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetString called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:  TYGetString(hDevice, componentID, featureID, buffer, bufferSize
	^ is invalid
	The hDevice parameter you input is not recorded
	Possible reasons:  1.TYOpenDevice failed to open device and get correct handle
	2.Memory in stack to store handle data is corrupted
	3.After getting handle, you updated device list by calling TYUpo
TY_STATUS_INVALID_COMPONENT	Invalid component ID
TI_STATOS_INVALID_COMFONENT	invalid component ib
	Suggestions:
	Please check componentID parameter Like this:
	TYGetString(hDevice, componentID, featureID, buffer, bufferSize
	^ is invalid
	componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the xi
	Tou dan also view the compensate of the damper by patalining the in
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:
	Please check featureID parameter
	Like this:
	TYGetString(hDevice, componentID, featureID, buffer, bufferSize
	You entered an invalid featureID parameter
	You can get a list of features of the camera device through TYGetl
	You can also view the features of the camera device by obtaining t
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:
	Please check the feature type
	Like this:
	TYGetString(hDevice, componentID, featureID, buffer, bufferSize ^ type mismatch
	The feature type you entered does not match. You can use TYFeature
TY_STATUS_NULL_POINTER	TYGetStringLength called with NULL pointer
	Suggestions:
	Please check your code
	Like this:  TYGetStringLength(hDevice, componentID, featureID, pLength);
	' is NULL

## See also

**TYGetString** 

# 5.1.2.40 TYGetStruct()

```
TY_CAPI TYGetStruct (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

void * pStruct,

uint32_t structSize )
```

#### Get value of struct.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	pStruct	Pointer of struct.
in	structSize	Size of input buffer pStruct.

#### Return values

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYGetStruct called with invalid device handle	
	Suggestions: Please check device handle Like this:	
	TYGetStruct(hDevice, componentID, featureID, pStr ^ is invalid The hDevice parameter you input is not recorded	ruct, structSize
	Possible reasons:	
	1.TYOpenDevice failed to open device and get corn 2.Memory in stack to store handle data is corrupt	ed
	3.After getting handle, you updated device list k	by calling TYUpo
TY_STATUS_INVALID_COMPONENT	Invalid component ID	
	Suggestions:  Please check componentID parameter Like this:  TYGetStruct(hDevice, componentID, featureID, pStr  ^ is invalid  componentID should be the value returned by TYGetCo	mponentIDs
TY_STATUS_INVALID_FEATURE	Invalid feature ID	
	Suggestions:  Please check featureID parameter  Like this:  TYGetStruct(hDevice, componentID, featureID, pStr	ruct, structSize
	You entered an invalid featureID parameter You can get a list of features of the camera device You can also view the features of the camera device	~
TY_STATUS_WRONG_TYPE	Feature type mismatch	
	Suggestions: Please check the feature type	

Like this:

TYGetStruct(hDevice, componentID, featureID, pStruct, structSize

The feature type you entered does not match. You can use TYFeature

^ type mismatch

## Return values

TY_STATUS_NULL_POINTER	TYGetStruct called with NULL pointer	
	Suggestions:  Please check your code  Like this:  TYGetStruct(hDevice, componentID, featureID, pSt  ^ i	ruct, structSize s NULL
TY_STATUS_WRONG_SIZE	Struct size mismatch	-
	Suggestions:  Please check the struct size Like this:  TYGetStruct(hDevice, componentID, featureID, pSt  The struct size you entered does not match	ruct, structSize ^ is inval
TY_STATUS_TIMEOUT	Failed to get struct feature	
	Suggestions: Possible reasons: 1.Network communication is abnormal, please chec	k whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get struct feature	]
	Suggestions: Possible reasons: 1.The feature of the camera device is not availa 2.Camera device is abnormal and cannot get struc	-

# 5.1.2.41 TYHasDevice()

Check whether the interface has the specified device.

#### **Parameters**

	in	ifaceHandle	Interface handle.
	in	deviceID	Device ID string, can be get from TY_DEVICE_BASE_INFO.
ĺ	out	value	True if the device exists.

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

TY_STATUS_INVALID_INTERFACE	TYHasDevice called with invalid interface handle	
	Suggestions:  Please check interface handle Like this:  TYHasDevice(ifaceHandle, deviceID, value);  ^ is invalid  The ifaceHandle parameter you input is not recorded Possible reasons:  1.TYOpenInterface failed to open interface and get 2.Memory in stack to store handle data is corrupte 3.After getting handle, you updated interface list	d
TY_STATUS_NULL_POINTER	TYHasDevice called with NULL pointer	
	Suggestions:  Please check your code Like this:  TYHasDevice(ifaceHandle, deviceID, value);  ^ or ^ is NULL	

## 5.1.2.42 TYHasFeature()

```
TY_CAPI TYHasFeature (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

bool * value )
```

Check whether a component has a specific feature.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	value	Whether has feature.

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYHasFeature called with invalid device handle
	Suggestions:  Please check device handle Like this:  TYHasFeature(hDevice, componentID, featureID, value);  ^ is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUR

## Return values

TY_STA	ATUS_INVALID_COMPONENT	Invalid component ID	
		Suggestions:	
		Please check componentID parameter	
		Like this:	
		TYHasFeature(hDevice, componentID, featureID, val ^ is invalid	.ue);
		componentID should be the value returned by TYGetCo	mponentIDs
		You can also view the components of the camera by	btaining the x
	TY_STATUS_NULL_POINTER	TYHasFeature called with NULL pointer	
		Suggestions:	
		Please check your code	
		Like this:	
		TYHasFeature(hDevice, componentID, featureID, va	ue);
		^ =	s NULL

# 5.1.2.43 TYHasInterface()

# Check if has interface.

## **Parameters**

in	ifaceID	Interface ID string, can be get from TY_INTERFACE_INFO.
out	value	True if the interface exists.

# Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	TYHasInterface called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYHasInterface(ifaceID, value);  ^ is NULL

# See also

TYGetInterfaceList

# 5.1.2.44 TYLibVersion()

Get current library version.

#### **Parameters**

out	version	Version infomation to be filled.
-----	---------	----------------------------------

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NULL_POINTER	TYLibVersion called with NULL pointer
	Suggestions:  Please check your code  Like this:  TYLibVersion(ver);  ^ is NULL

# 5.1.2.45 TYOpenDevice()

Open device by device ID.

#### **Parameters**

in	ifaceHandle	Interface handle.	
in	deviceID	Device ID string, can be get from TY_DEVICE_BASE_INFO.	
out	outDeviceHandle	Handle of opened device. Valid only if TY_STATUS_OK or TY_FW_ERRORCODE returned.	
out	outFwErrorcode	Firmware errorcode. Valid only if TY_FW_ERRORCODE returned.	

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

TY_STATUS_INVALID_INTERFACE	TYOpenDevice called with invalid interface handle
	Suggestions: Please check interface handle Like this:
	TYOpenDevice(ifaceHandle, deviceID, outDeviceHandle, outFwErrord
	The ifaceHandle parameter you input is not recorded  Possible reasons:
	<pre>1.TYOpenInterface failed to open interface and get correct handl 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling TY</pre>
TY_STATUS_NULL_POINTER	TYOpenDevice called with NULL pointer
	Suggestions: Please check your code Like this:
	TYOpenDevice(ifaceHandle, deviceID, outDeviceHandle, outFwErrord
TY_STATUS_INVALID_PARAMETER	TYOpenDevice called with invalid device ID: s
	Suggestions: Please check deviceID parameter Like this:
	TYOpenDevice(ifaceHandle, deviceID, outDeviceHandle, outFwErrorc
	Usually you get device information by calling TYUpdateDeviceList, and then open device by calling TYOpenDevice When your device online status changes); you may need to update device list again
TY_STATUS_BUSY	Failed to open device
	Suggestions: Possible reasons: 1.Camera is occupied, please check if other processes on this ma or other host machines are occupying the camera. If the camera 2.A third-party program is written into the camera, please conta
TY_STATUS_FIRMWARE_ERROR	Device opened successfully, but firmware error code is not 0
	Suggestions: Some functions of the device may have exceptions, please check the TY_FW_ERRORCODE outFwErrorcode; TYOpenDevice(ifaceHandle, deviceID, outDeviceHandle, &outFwErrorcoif(outFwErrorcode != 0) {    parse_firmware_errcode(outFwErrorcode); }
TY_STATUS_DEVICE_ERROR	Failed to open device
	Suggestions:  Possible reasons:  1.A third-party program is written into the camera, please conta  2.The camera IP address is not in the same network segment as th  If the camera IP address is not in the same network segment as  If there is a routing connection between your host and the cam  Otherwise, you can modify the camera IP address or the host IP  If you need to modify the camera IP address, please refer to S  3.Network communication is abnormal, please check whether the ne

# 5.1.2.46 TYOpenDeviceWithIP()

Open device by device IP, useful when a device is not listed.

#### **Parameters**

in	ifaceHandle	Interface handle.
in	IP	Device IP.
out	deviceHandle	Handle of opened device.

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY STATUS INVALID INTERFACE	TYOpenDeviceWithIP called with invalid interface handle
TI_STATOS_INVALID_INTERITACE	1 Topenbevicewithin called with invalid interlace handle
	Suggestions:
	Please check interface handle Like this:
	TYOpenDeviceWithIP(ifaceHandle, IP, outDeviceHandle);  ^ is invalid
	The ifaceHandle parameter you input is not recorded Possible reasons:
	1.TYOpenInterface failed to open interface and get correct hand
	2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling 3
TY_STATUS_NULL_POINTER	TYOpenDeviceWithIP called with NULL pointer
	Suggestions:
	Please check your code
	Like this:
	TYOpenDeviceWithIP(ifaceHandle, IP, outDeviceHandle);  ^ or^ is NULL
TY_STATUS_INVALID_PARAMETER	TYOpenDeviceWithIP called with invalid IP address
	Suggestions:
	Please check your code
	Like this:
	TYOpenDeviceWithIP(ifaceHandle, IP, outDeviceHandle);  ^ is invalid
	A valid IP address should be like: 192.168.31.1
	Usually you get device information by calling TYUpdateDeviceList,
	and then open device by calling TYOpenDevice
	When your device online status changes, you may need to update device list again
TY_STATUS_BUSY	Failed to open device
	Suggestions:
	Possible reasons:
	1. Camera is occupied, please check if other processes on this many than the company of the comp
	or other host machines are occupying the camera. If the camera. 2.A third-party program is written into the camera, please conf

## Return values

TY_STATUS_DEVICE_ERROR	Failed to open device	
	Suggestions: Possible reasons:	
	1.A third-party program is written into the camer 2.The camera IP address cannot communicate with t	_
	3.Network communication is abnormal, please check	

# 5.1.2.47 TYOpenInterface()

# Open specified interface.

## **Parameters**

in	ifaceID	Interface ID string, can be get from TY_INTERFACE_INFO.	
out	out outHandle Handle of opened interface.		

#### Return values

TY_STATUS_OK	Succeed.	
TY_STATUS_NOT_INITED	TYInitLib not called.	
TY_STATUS_NULL_POINTER	TYOpenInterface called with NULL pointer	
	Suggestions:  Please check your code  Like this:  TYOpenInterface(ifaceID, outHandle);  ^ or ^ is NULL	
TY_STATUS_INVALID_INTERFACE	TYOpenInterface called with invalid interface ID	
	Suggestions:  Please check ifaceID parameter  Like this:  TYOpenInterface(ifaceID, outHandle);  ^ is invalid  Usually you get interface information by calling TYU  and then open interface by calling TYOpenInterface  When your host interface (network or USB) changes);  you may need to update interface list again	pdateInterfaceL

### See also

TYGetInterfaceList

## 5.1.2.48 TYRegisterEventCallback()

Register device status callback. Register NULL to clean callback.

#### **Parameters**

in	hDevice	Device handle.
in	callback	Callback function.
in	userdata	User private data.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYRegisterEventCallback called with invalid device handle
	Suggestions:
	Please check device handle Like this:
	TYRegisterEventCallback(hDevice, callback, userdata);  ^ is invalid
	The hDevice parameter you input is not recorded
	1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted
	3.After getting handle, you updated device list by calling TYUpdat
TY_STATUS_BUSY	Device is capturing.

## 5.1.2.49 TYRegisterImuCallback()

```
TY_CAPI TYRegisterImuCallback (

TY_DEV_HANDLE hDevice,

TY_IMU_CALLBACK callback,

void * userdata )
```

Register imu callback. Register NULL to clean callback.

#### **Parameters**

in	hDevice	Device handle.
in	callback	Callback function.
in	userdata	User private data.

TY_STATUS_OK   Succeed.
-------------------------

## Return values

TY_STATUS_INVALID_HANDLE	TYRegisterImuCallback called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYRegisterImuCallback(hDevice, callback, userdata);	
	^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get correct	handle
	2.Memory in stack to store handle data is corrupted	
	3.After getting handle, you updated device list by ca	lling TYUpdateI
TY_STATUS_BUSY	Device is capturing.	

# 5.1.2.50 TYRemoveLogFile()

# Remove log file.

#### **Parameters**

in	filePath	Path to the log file.
----	----------	-----------------------

## Return values

TY_STATUS_OK	Succeed.	
TY_STATUS_ERROR	Failed to remove file	
	Suggestions: Please check if the file path is correct	

# 5.1.2.51 TYRemoveLogServer()

# Remove log server.

#### **Parameters**

in	protocol	Protocol of the server, "tcp" or "udp".	
in	ip	IP address of the server.	
in	port	Port of the server.	

TY_STATUS_OK	Succeed.
TY_STATUS_ERROR	Failed to remove server
	Suggestions: Please check if the ip and port are correct
TY_STATUS_INVALID_PARAMETER	Unsupported protocol
	Suggestions: Unsupported protocol, please use tcp or udp

# 5.1.2.52 TYSendSoftTrigger()

Send a software trigger to capture a frame when device works in trigger mode.

#### **Parameters**

in	hDevice	Device handle.
----	---------	----------------

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYSendSoftTrigger called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYSendSoftTrigger(hDevice);	
	^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	<ol> <li>TYOpenDevice failed to open device and get correct</li> </ol>	handle
	2. Memory in stack to store handle data is corrupted	
	3.After getting handle, you updated device list by c	alling TYUpdate
TY STATUS INVALID FEATURE	Not support soft trigger.	
TY_STATUS_IDLE	Camera device is not started	
	Suggestions:	
	Please start the camera device first	
	Like this:	
	TYStartCapture(hDevice);	
	TYSendSoftTrigger(hDevice);	
TY_STATUS_WRONG_MODE	Not in trigger mode.	
TY_STATUS_DEVICE_ERROR	Failed to send soft trigger	
	Suggestions:	
	Possible reasons:	
	<ol> <li>Camera device is abnormal and cannot send soft tri</li> <li>Network communication is abnormal, please check wh</li> </ol>	
	2. Network Communication is apploimat, prease check wh	ermer rme merwo

## Return values

TY_STATUS_BUSY	Failed to send soft trigger	
	Suggestions: Possible reasons: 1.Camera is busy, the last soft trigger is not comple	eted, please t

# 5.1.2.53 TYSetBool()

```
TY_CAPI TYSetBool (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

bool value )
```

#### Set value of bool feature.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
in	value	Bool value.

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYSetBool called with invalid device handle	
	Suggestions:  Please check device handle Like this:  TYSetBool(hDevice, componentID, featureID, value)  ^ is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get corn 2.Memory in stack to store handle data is corrupt 3.After getting handle, you updated device list k	rect handle
TY_STATUS_INVALID_COMPONENT	Invalid component ID	
	Suggestions:  Please check componentID parameter  Like this:  TYSetBool(hDevice, componentID, featureID, value)  ^ is invalid  componentID should be the value returned by TYGetCo	pmponentIDs

TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:  Please check featureID parameter Like this:  TYSetBool(hDevice, componentID, featureID, value);  ^ is invalid  You entered an invalid featureID parameter You can get a list of features of the camera device through TYGe You can also view the features of the camera device by obtaining
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type Like this:  TYSetBool(hDevice, componentID, featureID, value);  ^ type mismatch The feature type you entered does not match. You can use TYFeature
TY_STATUS_BUSY	Device is capturing, the feature is locked.
TY_STATUS_TIMEOUT	Failed to set bool feature
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the
TY_STATUS_DEVICE_ERROR	Failed to set bool feature
	Suggestions:  Possible reasons:  1.The feature of the camera device is not available or not imp 2.Camera device is abnormal and cannot set bool feature

# 5.1.2.54 TYSetByteArray()

```
TY_CAPI TYSetByteArray (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

const uint8_t * pBuffer,

uint32_t bufferSize )
```

Write byte array to device.

## **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
out	pBuffer	Byte buffer.
in	bufferSize	Size of buffer.

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYSetByteArray called with invalid device handle
	Suggestions:    Please check device handle    Like this:         TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffers
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:  Please check componentID parameter  Like this:  TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffers  ^ is invalid  componentID should be the value returned by TYGetComponentIDs  You can also view the components of the camera by obtaining the xr
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:  Please check featureID parameter  Like this:  TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffers  ^ is invalid  You entered an invalid featureID parameter  You can get a list of features of the camera device through TYGetI  You can also view the features of the camera device by obtaining the state of
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_NOT_PERMITTED  TY_STATUS_WRONG_TYPE	The feature is not writable.  Feature type mismatch
	Feature type mismatch  Suggestions: Please check the feature type Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffer: ^ type mismatch
TY_STATUS_WRONG_TYPE	Feature type mismatch  Suggestions: Please check the feature type Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffer: ^ type mismatch The feature type you entered does not match. You can use TYFeature  TYSetByteArray called with NULL pointer  Suggestions: Please check your code Like this:
TY_STATUS_WRONG_TYPE	Feature type mismatch  Suggestions: Please check the feature type Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffer, ^ type mismatch The feature type you entered does not match. You can use TYFeature  TYSetByteArray called with NULL pointer  Suggestions: Please check your code Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffer,
TY_STATUS_WRONG_TYPE  TY_STATUS_NULL_POINTER	Feature type mismatch  Suggestions: Please check the feature type Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffers ^ type mismatch The feature type you entered does not match. You can use TYFeature  TYSetByteArray called with NULL pointer  Suggestions: Please check your code Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffers) is NULL
TY_STATUS_WRONG_TYPE  TY_STATUS_NULL_POINTER	Feature type mismatch  Suggestions: Please check the feature type Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffer: ^ type mismatch The feature type you entered does not match. You can use TYFeature  TYSetByteArray called with NULL pointer  Suggestions: Please check your code Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffer: ^ is NULL  Array size mismatch  Suggestions: Please check the array size Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffer: ^ is in ^ is in
TY_STATUS_WRONG_TYPE  TY_STATUS_NULL_POINTER  TY_STATUS_WRONG_SIZE	Feature type mismatch  Suggestions: Please check the feature type Like this: TYSetByteArray(hDevice, componentID, featureID, pBuffer, buffers

TY_STATUS_DEVICE_ERROR	Failed to set byte array feature	
	Suggestions:  Possible reasons:  1.The feature of the camera device is not available 2.Camera device is abnormal and cannot set byte.	_

# 5.1.2.55 TYSetEnum()

```
TY_CAPI TYSetEnum (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

uint32_t value )
```

## Set value of enum feature.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
in	value	Enum value.

#### Return values

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYSetEnum called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYSetEnum(hDevice, componentID, featureID, value)	;
	^ is invalid	
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get corr	rect handle
	2.Memory in stack to store handle data is corrupt	ed
	3.After getting handle, you updated device list k	y calling TYUp
TY STATUS INVALID COMPONENT	Invalid component ID	
	·	
	Suggestions:	
	Please check componentID parameter	
	Like this:	
	TYSetEnum(hDevice, componentID, featureID, value) ^ is invalid	;

componentID should be the value returned by TYGetComponentIDs
You can also view the components of the camera by obtaining the x

## Return values

TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:  Please check featureID parameter Like this:  TYSetEnum(hDevice, componentID, featureID, value);  ^ is invalid  You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetl You can also view the features of the camera device by obtaining the state of the camera device
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type Like this:  TYSetEnum(hDevice, componentID, featureID, value);  ^ type mismatch  The feature type you entered does not match. You can use TYFeature
TY STATUS INVALID PARAMETER	Out of range
	Suggestions:  Please check the value Like this:  TYSetEnum(hDevice, componentID, featureID, value);  ^ is out of range The value is out of range, please use TYGetEnumEntryInfo to get the
TY_STATUS_BUSY	Device is capturing, the feature is locked.
TY_STATUS_TIMEOUT	Failed to set enum feature
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to set enum feature
	Suggestions: Possible reasons: 1.The feature of the camera device is not available or not imple 2.Camera device is abnormal and cannot set enum feature

# 5.1.2.56 TYSetFloat()

```
TY_CAPI TYSetFloat (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

float value )
```

# Set value of float feature.

## **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
in	value	Float value.

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYSetFloat called with invalid device handle
	Suggestions:
	Please check device handle
	<pre>Like this:     TYSetFloat(hDevice, componentID, featureID, value);</pre>
	The hDevice parameter you input is not recorded  Possible reasons:
	1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYU
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:
	Please check componentID parameter
	<pre>Like this:     TYSetFloat(hDevice, componentID, featureID, value);</pre>
	^ is invalid componentID should be the value returned by TYGetComponentIDs
	You can also view the components of the camera by obtaining the
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:
	Please check featureID parameter
	<pre>Like this:     TYSetFloat(hDevice, componentID, featureID, value);</pre>
	^ is invalid You entered an invalid featureID parameter
	You can get a list of features of the camera device through TYGe
	You can also view the features of the camera device by obtaining
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions: Please check the feature type
	Like this:
	TYSetFloat(hDevice, componentID, featureID, value);  ^ type mismatch
	The feature type you entered does not match. You can use TYFeatu
TY_STATUS_OUT_OF_RANGE	Out of range
_ <del>_</del> _	Suggestions:
	Please check the value
	Like this:  TYSetFloat(hDevice, componentID, featureID, value);
	^ is out of range The value is out of range, please use TYGetFloatRange to get the
TY_STATUS_BUSY	Device is capturing, the feature is locked.
TY_STATUS_TIMEOUT	Failed to set float feature
	Suggestions: Possible reasons:
	1.Network communication is abnormal, please check whether the
TY_STATUS_DEVICE_ERROR	Failed to set float feature
<del>_</del>	Suggestions:
	Possible reasons:
	1. The feature of the camera device is not available or not imp 2. Camera device is abnormal and cannot set float feature

## 5.1.2.57 TYSetInt()

```
TY_CAPI TYSetInt (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

int32_t value)
```

## Set value of integer feature.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
in	value	Integer value.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYSetInt called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:
	TYSetInt(hDevice, componentID, featureID, value);  ^ is invalid
	The hDevice parameter you input is not recorded
	Possible reasons:
	1.TYOpenDevice failed to open device and get correct handle
	2. Memory in stack to store handle data is corrupted
	3.After getting handle, you updated device list by calling TY
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:
	Please check componentID parameter
	Like this:
	TYSetInt(hDevice, componentID, featureID, value);  ^ is invalid
	componentID should be the value returned by TYGetComponentIDs
	You can also view the components of the camera by obtaining the
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:
	Please check featureID parameter
	Like this:
	TYSetInt(hDevice, componentID, featureID, value); ^ is invalid
	You entered an invalid featureID parameter
	You can get a list of features of the camera device through TYGe

You can also view the features of the camera device by obtaining

TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type Like this:  TYSetInt(hDevice, componentID, featureID, value);  ^ type mismatch  The feature type you entered does not match. You can use TYFeatur
TY_STATUS_OUT_OF_RANGE	Out of range
	Suggestions:  Please check the value  Like this:  TYSetInt(hDevice, componentID, featureID, value);  ^ is out of range  The value is out of range, please use TYGetIntRange to get the ra
TY_STATUS_BUSY	Device is capturing, the feature is locked.
TY_STATUS_TIMEOUT	Failed to set int feature
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the n
TY_STATUS_DEVICE_ERROR	Failed to set int feature
	Suggestions: Possible reasons: 1.The feature of the camera device is not available or not impl 2.Camera device is abnormal and cannot set int feature

# 5.1.2.58 TYSetLogLevel()

```
TY_CAPI TYSetLogLevel ( {\tt TY\_LOG\_LEVEL}\ lvl\ )
```

# Set log level.

## **Parameters**

in	lvl	Log level.

## Return values

```
TY_STATUS_OK Succeed.
```

# 5.1.2.59 TYSetLogPrefix()

# set log prefix

#### **Parameters**

in	prefix	Prefix string.	
in	prefix	Prefix string.	

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_PARAMETER	Prefix is empty or prefix is too long
	Suggestions:  Prefix is empty or prefix is too long, cannot be set Like this:  TYSetLogPrefix(prefix);  ^ prefix is empty or prefix is too

# 5.1.2.60 TYSetString()

```
TY_CAPI TYSetString (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,

const char * buffer )
```

# Set value of string feature.

#### **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
in	buffer	String buffer.

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYSetString called with invalid device handle	
	Suggestions:	
	Please check device handle	
	Like this:	
	TYSetString(hDevice, componentID, featureID, pBuffe ^ is invalid	er);
	The hDevice parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenDevice failed to open device and get correct	ct handle
	2.Memory in stack to store handle data is corrupted	d
	3.After getting handle, you updated device list by	calling T

TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:  Please check componentID parameter Like this:  TYSetString(hDevice, componentID, featureID, pBuffer);  ^ is invalid  componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the second
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:  Please check featureID parameter Like this:  TYSetString(hDevice, componentID, featureID, pBuffer);  ^ is invalid  You entered an invalid featureID parameter You can get a list of features of the camera device through TYGet You can also view the features of the camera device by obtaining
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type Like this:  TYSetString(hDevice, componentID, featureID, pBuffer);  ^ type mismatch  The feature type you entered does not match. You can use TYFeature
TY_STATUS_NULL_POINTER	TYSetString called with NULL pointer
	Suggestions:    Please check your code    Like this:     TYSetString(hDevice, componentID, featureID, pBuffer);
TY_STATUS_OUT_OF_RANGE	Input string is too long.
TY_STATUS_BUSY	Device is capturing, the feature is locked.
TY_STATUS_TIMEOUT	Failed to set string feature
TY_STATUS_DEVICE_ERROR	Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the related to set string feature
	Suggestions:  Possible reasons:  1.The feature of the camera device is not available or not impleated and cannot set string feature

# 5.1.2.61 TYSetStruct()

```
TY_CAPI TYSetStruct (

TY_DEV_HANDLE hDevice,

TY_COMPONENT_ID componentID,

TY_FEATURE_ID featureID,
```

void \* pStruct, uint32\_t structSize )

TY\_STATUS\_OK Succeed.

#### Set value of struct.

## **Parameters**

in	hDevice	Device handle.
in	componentID	Component ID.
in	featureID	Feature ID.
in	pStruct	Pointer of struct.
in	structSize	Size of struct.

11_0111100_011	Guocou.
TY_STATUS_INVALID_HANDLE	TYSetStruct called with invalid device handle
	Suggestions:  Please check device handle Like this:  TYSetStruct(hDevice, componentID, featureID, pStruct, structSize  ^ is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUpe
TY_STATUS_INVALID_COMPONENT	Invalid component ID
	Suggestions:  Please check componentID parameter  Like this:  TYSetStruct(hDevice, componentID, featureID, pStruct, structSize  ^ is invalid  componentID should be the value returned by TYGetComponentIDs  You can also view the components of the camera by obtaining the xr
TY_STATUS_INVALID_FEATURE	Invalid feature ID
	Suggestions:  Please check featureID parameter Like this:  TYSetStruct(hDevice, componentID, featureID, pStruct, structSize  ^ is invalid You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetI You can also view the features of the camera device by obtaining the struct of the struct of the camera device by obtaining the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct of the struct
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	Feature type mismatch
	Suggestions:  Please check the feature type  Like this:  TYSetStruct(hDevice, componentID, featureID, pStruct, structSize  ^ type mismatch  The feature type you entered does not match. You can use TYFeature

TY_STATUS_NULL_POINTER	TYSetStruct called with NULL pointer	
	Suggestions:  Please check your code Like this:  TYSetStruct(hDevice, componentID, featureID, pSt.	ruct, structSiza NULL
TY STATUS WRONG SIZE	Struct size mismatch	
TY STATUS BUSY	Suggestions:     Please check the struct size     Like this:     TYSetStruct(hDevice, componentID, featureID, pStruct size you entered does not match  Device is capturing, the feature is locked.	ruct, structSize ^ is inva
TY STATUS TIMEOUT	Failed to set struct feature	
	Suggestions: Possible reasons: 1.Network communication is abnormal, please check	whether the ne
TY_STATUS_DEVICE_ERROR	Failed to set struct feature	
	Suggestions: Possible reasons: 1.The feature of the camera device is not availal 2.Camera device is abnormal and cannot set struct	_

# 5.1.2.62 TYStartCapture()

# Start capture.

#### **Parameters**

TY_STATUS_OK	Succeed.	
TY_STATUS_INVALID_HANDLE	TYStartCapture called with invalid device handle	
	Suggestions:  Please check device handle Like this:  TYStartCapture(hDevice);  ^ is invalid  The hDevice parameter you input is not recorded Possible reasons:  1.TYOpenDevice failed to open device and get corn 2.Memory in stack to store handle data is corrupt 3.After getting handle, you updated device list k	ed

## Return values

TY_STATUS_INVALID_COMPONENT	No components are enabled		
	Suggestions:  Please enable the components of the camera device for Like this:  TYEnableComponents(hDevice, componentIDs);  TYStartCapture(hDevice);	first	
TY_STATUS_BUSY	Camera device has been started.		
	Suggestions:  Please stop the camera device first Like this:  TYStopCapture(hDevice);  TYStartCapture(hDevice);		
TY_STATUS_DEVICE_ERROR	Failed to start camera		
	Suggestions: Possible reasons: 1.Camera device is abnormal and cannot start the 2.Network communication is abnormal, please check 3.Camera is busy, please try again		:he n

# 5.1.2.63 TYStopCapture()

# Stop capture.

#### **Parameters**

in	hDevice	Device handle.
----	---------	----------------

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYStopCapture called with invalid device handle
	Suggestions:
	Please check device handle
	Like this:
	TYStopCapture(hDevice);
	^ is invalid
	The hDevice parameter you input is not recorded
	Possible reasons:
	1.TYOpenDevice failed to open device and get correct
	2.Memory in stack to store handle data is corrupted
	3.After getting handle, you updated device list by ca

## Return values

TY_STATUS_IDLE	Camera device has been stopped	
	Suggestions:  The camera device has stopped, usually after starting Like this:  TYStartCapture(hDevice);  TYStopCapture(hDevice);	
TY_STATUS_DEVICE_ERROR	Failed to stop camera	
	Suggestions: Possible reasons: 1.Camera device is abnormal and cannot stop the camera	

## 5.1.2.64 TYUpdateAllDeviceList()

Update current connected devices.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

## 5.1.2.65 TYUpdateDeviceList()

Update current connected devices.

#### **Parameters**

ı			
	in	ifaceHandle	Interface handle.

## Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

## Return values

TY_STATUS_INVALID_INTERFACE	TYUpdateDeviceList called with invalid interface handle	
	Suggestions:	
	Please check interface handle	
	Like this:	
	TYUpdateDeviceList(ifaceHandle);	
	^ is invalid	
	The ifaceHandle parameter you input is not recorded	
	Possible reasons:	
	1.TYOpenInterface failed to open interface and get co	orrect handle
	2.Memory in stack to store handle data is corrupted	
	3.After getting handle, you updated interface list by	y calling TYU

## 5.1.2.66 TYUpdateInterfaceList()

Update current interfaces. call before TYGetInterfaceList.

#### Return values

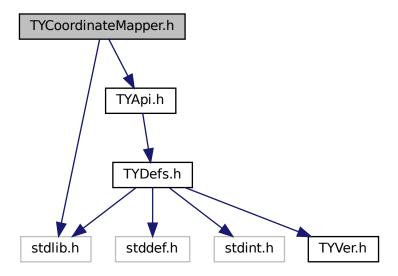
TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

# 5.2 TYCoordinateMapper.h File Reference

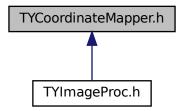
Coordinate Conversion API.

```
#include <stdlib.h>
#include "TYApi.h"
```

Include dependency graph for TYCoordinateMapper.h:



This graph shows which files directly or indirectly include this file:



## Classes

- struct TY\_PIXEL\_DESC
- struct TY\_PIXEL\_COLOR\_DESC

#### **Macros**

• #define **TYMAP\_CHECKRET**(f, bufToFree)

## **Typedefs**

- typedef struct TY PIXEL DESC TY PIXEL DESC
- typedef struct TY PIXEL COLOR DESC TY PIXEL COLOR DESC

#### **Functions**

• TY\_CAPI TYInvertExtrinsic (const TY\_CAMERA\_EXTRINSIC \*orgExtrinsic, TY\_CAMERA\_EXTRINSIC \*invExtrinsic)

Calculate 4x4 extrinsic matrix's inverse matrix.

TY\_CAPI TYMapDepthToPoint3d (const TY\_CAMERA\_CALIB\_INFO \*src\_calib, uint32\_t depthW, uint32
 \_t depthH, const TY\_PIXEL\_DESC \*depthPixels, uint32\_t count, TY\_VECT\_3F \*point3d, float f\_scale\_
 unit=1.0f)

Map pixels on depth image to 3D points.

TY\_CAPI TYMapPoint3dToDepth (const TY\_CAMERA\_CALIB\_INFO \*dst\_calib, const TY\_VECT\_3F \*point3d, uint32\_t count, uint32\_t depthW, uint32\_t depthH, TY\_PIXEL\_DESC \*depth, float f\_scale\_
 unit=1.0f)

Map 3D points to pixels on depth image. Reverse operation of TYMapDepthToPoint3d.

• TY\_CAPI TYMapDepthImageToPoint3d (const TY\_CAMERA\_CALIB\_INFO \*src\_calib, int32\_t imageW, int32\_t imageH, const uint16\_t \*depth, TY\_VECT\_3F \*point3d, float f\_scale\_unit=1.0f)

Map depth image to 3D points. 0 depth pixels maps to (NAN, NAN, NAN).

- TY\_CAPI TYDepthImageFillEmptyRegion (uint16\_t \*depth, uint32\_t depthW, uint32\_t depthH) Fill depth image empty region.
- TY\_CAPI TYMapPoint3dToDepthImage (const TY\_CAMERA\_CALIB\_INFO \*dst\_calib, const TY\_VECT\_3F \*point3d, uint32\_t count, uint32\_t depthW, uint32\_t depthH, uint16\_t \*depth, float f\_target\_scale=1.0f)

Map 3D points to depth image. (NAN, NAN, NAN) will be skipped.

TY\_CAPI TYMapPoint3dToPoint3d (const TY\_CAMERA\_EXTRINSIC \*extrinsic, const TY\_VECT\_3F \*point3dFrom, int32\_t count, TY\_VECT\_3F \*point3dTo)

Map 3D points to another coordinate.

void TYPixelsOverlapRemove (TY\_PIXEL\_DESC \*lut, uint32\_t count, uint32\_t imageW, uint32\_t imageH)

## 5.2.1 Detailed Description

Coordinate Conversion API.

Note

Considering performance, we leave the responsibility of parameters check to users.

Copyright

Copyright(C)2016-2018 Percipio All Rights Reserved

#### 5.2.2 Macro Definition Documentation

## 5.2.2.1 TYMAP\_CHECKRET

Definition at line 274 of file TYCoordinateMapper.h.

## 5.2.3 Function Documentation

## 5.2.3.1 TYDepthImageFillEmptyRegion()

Fill depth image empty region.

## Parameters

	in	depth	Depth image pixels.
ĺ	in	depthW	Width of current depth image.
ĺ	in	depthH	Height of current depth image.

#### 5.2.3.2 TYInvertExtrinsic()

Calculate 4x4 extrinsic matrix's inverse matrix.

#### **Parameters**

in	orgExtrinsic	Input extrinsic matrix.
out	invExtrinsic	Inverse matrix.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_ERROR	Calculation failed.

## 5.2.3.3 TYMapDepthImageToPoint3d()

Map depth image to 3D points. 0 depth pixels maps to (NAN, NAN, NAN).

#### **Parameters**

in	src_calib	Depth image's calibration data.
in	depthW	Width of depth image.
in	depthH	Height of depth image.
in	depth	Depth image.
out	point3d	Output point3D image.

#### Return values

TY_STATUS_OK	Succeed.
--------------	----------

## 5.2.3.4 TYMapDepthToPoint3d()

Map pixels on depth image to 3D points.

#### **Parameters**

in	src_calib	Depth image's calibration data.
in	depthW	Width of depth image.

#### **Parameters**

in	depthH	Height of depth image.
in	depthPixels	Pixels on depth image.
in	count	Number of depth pixels.
out	point3d	Output point3D.

#### Return values

TY_STATUS_OK	Succeed.
--------------	----------

#### 5.2.3.5 TYMapPoint3dToDepth()

Map 3D points to pixels on depth image. Reverse operation of TYMapDepthToPoint3d.

#### **Parameters**

in	dst_calib	Target depth image's calibration data.
in	point3d	Input 3D points.
in	count	Number of points.
in	depthW	Width of target depth image.
in	depthH	Height of target depth image.
out	depth	Output depth pixels.

#### **Return values**

```
TY_STATUS_OK Succeed.
```

## 5.2.3.6 TYMapPoint3dToDepthImage()

```
uint32_t depthH,
uint16_t * depth,
float f_target_scale = 1.0f )
```

Map 3D points to depth image. (NAN, NAN, NAN) will be skipped.

#### **Parameters**

in	dst_calib	Target depth image's calibration data.
in	point3d	Input 3D points.
in	count	Number of points.
in	depthW	Width of target depth image.
in	depthH	Height of target depth image.
in,out	depth	Depth image buffer.

#### Return values

TY_STATUS_OK	Succeed.
--------------	----------

#### 5.2.3.7 TYMapPoint3dToPoint3d()

Map 3D points to another coordinate.

#### **Parameters**

in	extrinsic	Extrinsic matrix.
in	point3dFrom	Source 3D points.
in	count	Number of source 3D points.
out	point3dTo	Target 3D points.

#### Return values

```
TY_STATUS_OK Succeed.
```

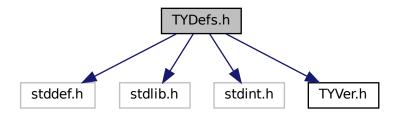
## 5.3 TYDefs.h File Reference

TYDefs.h includes camera control and data receiving data definitions which supports configuration for image resolution, frame rate, exposure time, gain, working mode,etc.

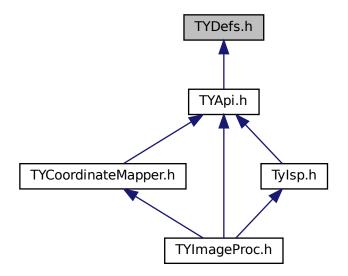
5.3 TYDefs.h File Reference 109

```
#include <stddef.h>
#include <stdlib.h>
#include <stdint.h>
#include "TYVer.h"
```

Include dependency graph for TYDefs.h:



This graph shows which files directly or indirectly include this file:



#### **Classes**

- struct TY\_VERSION\_INFO
- struct TY\_DEVICE\_NET\_INFO

device network information

- struct TY\_DEVICE\_USB\_INFO
- struct TY\_INTERFACE\_INFO
- struct TY\_DEVICE\_BASE\_INFO

- struct TY\_FEATURE\_INFO
- struct TY\_INT\_RANGE
- struct TY\_FLOAT\_RANGE

float range data structure

struct TY BYTEARRAY ATTR

byte array data structure

- struct TY ENUM ENTRY
- struct TY\_VECT\_3F
- struct TY CAMERA INTRINSIC
- struct TY\_CAMERA\_EXTRINSIC
- struct TY\_CAMERA\_DISTORTION
- struct TY\_CAMERA\_CALIB\_INFO
- struct TY\_TRIGGER\_PARAM
- struct TY TRIGGER PARAM EX
- struct TY\_TRIGGER\_TIMER\_LIST
- struct TY\_TRIGGER\_TIMER\_PERIOD
- struct TY\_AEC\_ROI\_PARAM
- struct TY\_PHC\_GROUP\_ATTR
- · struct TY PHC GROUP ATTR::phc group attr
- struct pattern\_sine\_param
- · struct pattern\_gray\_param
- struct pattern\_bin\_param
- struct TY\_LASER\_PATTERN\_PARAM
- struct TY CAMERA STATISTICS
- struct TY\_IMU\_DATA
- struct TY\_ACC\_BIAS
- struct TY\_ACC\_MISALIGNMENT
- struct TY\_ACC\_SCALE
- struct TY\_GYRO\_BIAS
- struct TY\_GYRO\_MISALIGNMENT
- struct TY\_GYRO\_SCALE
- struct TY\_CAMERA\_TO\_IMU
- struct TY\_TOF\_FREQ
- struct TY LASER PARAM
- struct TY IMAGE DATA
- struct TY\_FRAME\_DATA
- struct TY EVENT INFO
- struct TY\_DO\_WORKMODE
- struct TY\_DI\_WORKMODE
- struct TY\_TEMP\_DATA
- struct TY\_CAMERA\_ROTATION

#### **Macros**

- #define STDBOOL H
- #define \_\_bool\_true\_false\_are\_defined 1
- #define bool \_Bool
- #define true 1
- #define false 0
- #define TY\_DLLIMPORT \_\_attribute\_\_((visibility("default")))
- #define TY\_DLLEXPORT \_\_attribute\_\_((visibility("default")))
- · #define TY STDC
- #define TY\_CDEC

- #define TY\_EXPORT TY\_DLLIMPORT
- #define TY\_EXTC
- #define TY\_CAPI TY\_EXTC TY\_EXPORT TY\_STATUS TY\_STDC
- · #define TY\_INT\_SGBM\_COST\_PARAMITY\_INT\_SGBM\_UNIQUE\_MAX\_COST
- #define TY BOOL FLASHLIGHT TY BOOL IR FLASHLIGHT
- #define TY\_INT\_FLASHLIGHT\_INTENSITY TY\_INT\_IR\_FLASHLIGHT\_INTENSITY
- #define TY\_INT\_AE\_TARGET\_V TY\_INT\_AE\_TARGET\_Y
- #define TY\_DECLARE\_IMAGE\_MODE1(pix)

#### **Typedefs**

typedef enum TY\_STATUS\_LIST TY\_STATUS\_LIST

API call return status.

- typedef int32\_t TY\_STATUS
- typedef enum TY\_FW\_ERRORCODE\_LIST TY\_FW\_ERRORCODE\_LIST
- typedef uint32\_t TY\_FW\_ERRORCODE
- typedef enum TY EVENT LIST TY\_ENENT\_LIST
- typedef int32 t TY EVENT
- typedef void \* TY INTERFACE HANDLE

Interface handle.

typedef void \* TY\_DEV\_HANDLE

Device Handle.

- typedef enum TY\_DEVICE\_COMPONENT\_LIST TY\_DEVICE\_COMPONENT\_LIST
- typedef uint32 t TY COMPONENT ID

component unique id

· typedef enum TY FEATURE TYPE LIST TY FEATURE TYPE LIST

Feature Format Type definitions.

- typedef uint32\_t TY\_FEATURE\_TYPE
- typedef enum TY\_FEATURE\_ID\_LIST TY\_FEATURE\_ID\_LIST

feature for component definitions

typedef uint32\_t TY\_FEATURE\_ID

feature unique id

- typedef enum TY\_CONFIG\_MODE\_LIST TY\_CONFIG\_MODE\_LIST
- typedef uint32\_t TY\_CONFIG\_MODE
- typedef enum TY\_DEPTH\_QUALITY\_LIST TY\_DEPTH\_QUALITY\_LIST
- typedef uint32\_t TY\_DEPTH\_QUALITY
- typedef enum TY\_TRIGGER\_POL\_LIST TY\_TRIGGER\_POL\_LIST

set external trigger signal edge

- typedef uint32\_t TY\_TRIGGER\_POL
- typedef enum TY\_INTERFACE\_TYPE\_LIST TY\_INTERFACE\_TYPE\_LIST
- typedef uint32\_t TY\_INTERFACE\_TYPE
- typedef enum TY\_ACCESS\_MODE\_LIST TY\_ACCESS\_MODE\_LIST
- typedef uint8\_t TY\_ACCESS\_MODE
- typedef enum TY\_STREAM\_ASYNC\_MODE\_LIST TY\_STREAM\_ASYNC\_MODE\_LIST

stream async mode

- typedef uint8\_t TY\_STREAM\_ASYNC\_MODE
- typedef enum TY\_PIXEL\_BITS\_LIST TY\_PIXEL\_BITS\_LIST
- typedef uint32\_t TY\_PIXEL\_BITS
- typedef enum TY\_PIXEL\_FORMAT\_LIST TY\_PIXEL\_FORMAT\_LIST

pixel format definitions

- typedef uint32\_t TY\_PIXEL\_FORMAT
- typedef enum TY\_RESOLUTION\_MODE\_LIST TY\_RESOLUTION\_MODE\_LIST

predefined resolution list

- typedef int32 t TY RESOLUTION MODE
- typedef enum TY IMAGE MODE LIST TY IMAGE MODE LIST

Predefined Image Mode List image mode controls image resolution & format predefined image modes named like TY\_IMAGE\_MODE\_MONO\_160x120,TY\_IMAGE\_MODE\_RGB\_1280x960.

- typedef uint32\_t TY\_IMAGE\_MODE
- typedef enum TY TRIGGER MODE LIST TY TRIGGER MODE LIST
- typedef int16 t TY\_TRIGGER\_MODE
- typedef enum TY\_TIME\_SYNC\_TYPE\_LIST TY\_TIME\_SYNC\_TYPE\_LIST

type of time sync

- typedef uint32 t TY TIME SYNC TYPE
- typedef uint32 t TY E VOLT T
- typedef uint32 t TY E DO MODE
- typedef uint32\_t TY\_E\_DI\_MODE
- typedef uint32\_t TY\_E\_DI\_INT\_ACTION
- typedef uint32 t TY\_TEMPERATURE\_ID
- typedef enum TY LOG LEVEL LIST TY LOG LEVEL LIST
- typedef int32 t TY LOG LEVEL
- typedef struct TY\_VERSION\_INFO TY\_VERSION\_INFO
- typedef struct TY\_DEVICE\_NET\_INFO TY\_DEVICE\_NET\_INFO

device network information

- typedef struct TY DEVICE USB INFO TY DEVICE USB INFO
- typedef struct TY INTERFACE INFO TY INTERFACE INFO
- typedef struct TY\_DEVICE\_BASE\_INFO TY\_DEVICE\_BASE\_INFO
- typedef enum TY\_VISIBILITY\_TYPE TY\_VISIBILITY\_TYPE
- typedef struct TY\_FEATURE\_INFO TY\_FEATURE\_INFO
- typedef struct TY\_INT\_RANGE TY\_INT\_RANGE
- typedef struct TY\_FLOAT\_RANGE TY\_FLOAT\_RANGE

float range data structure

typedef struct TY\_BYTEARRAY\_ATTR TY\_BYTEARRAY\_ATTR

byte array data structure

- typedef struct TY\_ENUM\_ENTRY TY\_ENUM\_ENTRY
- typedef struct TY\_VECT\_3F TY\_VECT\_3F
- typedef struct TY\_CAMERA\_INTRINSIC TY\_CAMERA\_INTRINSIC
- typedef struct TY CAMERA EXTRINSIC TY CAMERA EXTRINSIC
- typedef struct TY\_CAMERA\_DISTORTION TY\_CAMERA\_DISTORTION
- typedef struct TY\_CAMERA\_CALIB\_INFO TY\_CAMERA\_CALIB\_INFO
- typedef struct TY\_TRIGGER\_PARAM TY\_TRIGGER\_PARAM
- typedef struct TY\_TRIGGER\_PARAM\_EX TY\_TRIGGER\_PARAM\_EX
- typedef struct TY\_TRIGGER\_TIMER\_LIST TY\_TRIGGER\_TIMER\_LIST
- typedef struct TY\_TRIGGER\_TIMER\_PERIOD TY\_TRIGGER\_TIMER\_PERIOD
- typedef struct TY AEC ROI PARAM TY AEC ROI PARAM
- typedef struct TY PHC GROUP ATTR TY PHC GROUP ATTR
- typedef struct TY LASER PATTERN PARAM TY LASER PATTERN PARAM
- typedef struct TY CAMERA STATISTICS TY CAMERA STATISTICS
- typedef struct TY IMU DATA TY IMU DATA
- typedef struct TY\_ACC\_BIAS TY\_ACC\_BIAS
- typedef struct TY\_ACC\_MISALIGNMENT TY\_ACC\_MISALIGNMENT
- typedef struct TY ACC SCALE TY ACC SCALE
- typedef struct TY\_GYRO\_BIAS TY\_GYRO\_BIAS
- typedef struct TY GYRO MISALIGNMENT TY GYRO MISALIGNMENT
- typedef struct TY\_GYRO\_SCALE TY\_GYRO\_SCALE

5.3 TYDefs.h File Reference 113

- typedef struct TY\_CAMERA\_TO\_IMU TY\_CAMERA\_TO\_IMU
- typedef struct TY\_TOF\_FREQ TY\_TOF\_FREQ
- typedef enum TY\_IMU\_FPS\_LIST TY\_IMU\_FPS\_LIST
- typedef struct TY LASER PARAM TY\_LASER\_PARAM
- typedef struct TY IMAGE DATA TY IMAGE DATA
- typedef struct TY FRAME DATA TY FRAME DATA
- typedef struct TY EVENT INFO TY EVENT INFO
- typedef struct TY DO WORKMODE TY DO WORKMODE
- typedef struct TY\_DI\_WORKMODE TY\_DI\_WORKMODE
- typedef struct TY\_TEMP\_DATA TY\_TEMP\_DATA
- typedef struct TY\_CAMERA\_ROTATION TY\_CAMERA\_ROTATION

#### **Enumerations**

enum TY STATUS LIST: int32 t { TY STATUS OK = 0, TY STATUS ERROR = -1001, TY STATUS NOT INITED = -1002, TY STATUS ↔ **NOT\_IMPLEMENTED** = -1003, TY STATUS NOT PERMITTED = -1004, TY STATUS DEVICE ERROR = -1005, TY STATUS INVA↔ LID PARAMETER = -1006, TY STATUS INVALID HANDLE = -1007, TY\_STATUS\_INVALID\_COMPONENT = -1008, TY\_STATUS\_INVALID\_FEATURE = -1009, TY\_STATU S\_WRONG\_TYPE = -1010, TY\_STATUS\_WRONG\_SIZE = -1011, TY\_STATUS\_OUT\_OF\_MEMORY = -1012, TY\_STATUS\_OUT\_OF\_RANGE = -1013, TY\_STATUS\_TIM ← EOUT = -1014, TY STATUS WRONG MODE = -1015, TY STATUS BUSY = -1016, TY STATUS IDLE = -1017, TY STATUS NO DATA = -1018, TY STATU ← S NO BUFFER = -1019, TY STATUS NULL POINTER = -1020, TY STATUS READONLY FEATURE = -1021, TY STATUS I↔ NVALID DESCRIPTOR = -1022, TY STATUS INVALID INTERFACE = -1023, TY\_STATUS\_FIRMWARE\_ERROR = -1024, TY\_STATUS\_DEV\_EPERM = -1, TY\_STATUS\_DEV\_EIO = -5, TY STATUS DEV ENOMEM = -12, TY STATUS DEV EBUSY = -16, TY STATUS DEV EINVAL = -22 }

API call return status.

enum TY\_FW\_ERRORCODE\_LIST: uint32\_t {

TY\_FW\_ERRORCODE\_CAM0\_NOT\_DETECTED = 0x000000001, TY\_FW\_ERRORCODE\_CAM1\_NOT\_ $\hookleftarrow$  DETECTED = 0x000000002, TY\_FW\_ERRORCODE\_CAM2\_NOT\_DETECTED = 0x000000004, TY\_FW\_E $\hookleftarrow$  RRORCODE POE NOT INIT = 0x000000008.

TY\_FW\_ERRORCODE\_RECMAP\_NOT\_CORRECT = 0x00000010, TY\_FW\_ERRORCODE\_LOOKUPT  $\leftarrow$  ABLE\_NOT\_CORRECT = 0x00000020, TY\_FW\_ERRORCODE\_DRV8899\_NOT\_INIT = 0x00000040, T $\leftarrow$  Y\_FW\_ERRORCODE\_FOC\_START\_ERR = 0x00000080,

TY\_FW\_ERRORCODE\_CONFIG\_NOT\_FOUND = 0x00010000, TY\_FW\_ERRORCODE\_CONFIG\_NOT ← CORRECT = 0x00020000, TY\_FW\_ERRORCODE\_XML\_NOT\_FOUND = 0x00040000, TY\_FW\_ERRO ← RCODE XML NOT CORRECT = 0x00080000,

TY\_FW\_ERRORCODE\_XML\_OVERRIDE\_FAILED = 0x00100000, TY\_FW\_ERRORCODE\_CAM\_INIT\_←
FAILED = 0x00200000, TY\_FW\_ERRORCODE\_LASER\_INIT\_FAILED = 0x00400000 }

- enum TY\_EVENT\_LIST: int32\_t { TY\_EVENT\_DEVICE\_OFFLINE = -2001, TY\_EVENT\_LICENSE\_ERR ← OR = -2002, TY\_EVENT\_FW\_INIT\_ERROR = -2003 }
- enum TY DEVICE COMPONENT LIST: uint32 t {

TY\_COMPONENT\_DEVICE = 0x80000000, TY\_COMPONENT\_DEPTH\_CAM = 0x00010000, TY\_COMPONENT\_IR\_CAM\_LI = 0x00040000, TY\_COMPONENT\_IR\_CAM\_RIGHT = 0x00080000,

TY\_COMPONENT\_LASER = 0x00400000, TY\_COMPONENT\_IMU = 0x00800000,

TY\_COMPONENT\_BRIGHT\_HISTO = 0x01000000, TY\_COMPONENT\_STORAGE = 0x02000000, TY\_COMPONENT\_RGB\_CAM = TY\_COMPONENT\_RGB\_CAM\_LEFT }

enum TY\_FEATURE\_TYPE\_LIST : uint32\_t {

TY\_FEATURE\_INT = 0x1000, TY\_FEATURE\_FLOAT = 0X2000, TY\_FEATURE\_ENUM = 0x3000, TY\_F $\leftrightarrow$  EATURE BOOL = 0x4000.

TY\_FEATURE\_STRING = 0x5000, TY\_FEATURE\_BYTEARRAY = 0x6000, TY\_FEATURE\_STRUCT = 0x7000 }

Feature Format Type definitions.

ASER\_LUT\_IDX = 0x050a | TY FEATURE INT,

enum TY FEATURE ID LIST: uint32 t { TY\_STRUCT\_CAM\_INTRINSIC = 0x0000 | TY\_FEATURE\_STRUCT, TY\_STRUCT\_EXTRINSIC\_TO\_DEPTH = 0x0001 | TY\_FEATURE\_STRUCT, TY\_STRUCT\_EXTRINSIC\_TO\_IR\_LEFT = 0x0002 | TY\_FEATURE ↔ STRUCT, TY STRUCT CAM RECTIFIED ROTATION = 0x0003 | TY FEATURE STRUCT, TY\_STRUCT\_CAM\_DISTORTION = 0x0006 | TY\_FEATURE\_STRUCT, TY\_STRUCT\_CAM\_CALIB\_DATA = 0x0007 | TY\_FEATURE\_STRUCT, TY\_STRUCT\_CAM\_RECTIFIED\_INTRI = 0x0008 | TY\_FEATURE\_← STRUCT, TY\_BYTEARRAY\_CUSTOM\_BLOCK = 0x000A | TY\_FEATURE\_BYTEARRAY, TY BYTEARRAY ISP BLOCK = 0x000B | TY FEATURE BYTEARRAY, TY INT PERSISTENT IP = 0x0010 | TY FEATURE INT, TY INT PERSISTENT SUBMASK = 0x0011 | TY FEATURE INT, TY IN↔ T PERSISTENT GATEWAY = 0x0012 | TY FEATURE INT, TY BOOL GVSP RESEND = 0x0013 | TY FEATURE BOOL, TY INT PACKET DELAY = 0x0014 | TY -FEATURE INT, TY INT ACCEPTABLE PERCENT = 0x0015 | TY FEATURE INT, TY INT NTP SERVER IP = 0x0016 | TY FEATURE INT, TY\_INT\_PACKET\_SIZE = 0x0017 | TY\_FEATURE\_INT, TY\_INT\_LINK\_CMD\_TIMEOUT = 0x0018 | TY\_F↔ EATURE INT, TY STRUCT CAM STATISTICS = 0x00ff | TY FEATURE STRUCT, TY INT WIDTH MAX = 0x0100 | TY FEATURE INT, TY INT HEIGHT MAX = 0x0101 | TY FEATURE INT, TY INT OFFSET X = 0x0102 | TY FEATURE INT, TY INT OFFSET Y = 0x0103 | TY FEATURE INT, TY INT WIDTH = 0x0104 | TY FEATURE INT, TY INT HEIGHT = 0x0105 | TY FEATURE INT, TY ENUM IMAGE MODE = 0x0109 | TY FEATURE ← ENUM, TY FLOAT SCALE UNIT = 0x010a | TY FEATURE FLOAT, TY ENUM TRIGGER POL = 0x0201 TY FEATURE ENUM. TY INT FRAME PER TRIGGER = 0x0202 | TY FEATURE INT, TY STRUCT TRIGGER PARAM = 0x0523 | TY FEATURE STRUCT, TY STRUCT TRIGGER PARAM EX = 0x0525 | TY FEATURE ST ← RUCT, TY STRUCT TRIGGER TIMER LIST = 0x0526 | TY FEATURE STRUCT, TY\_STRUCT\_TRIGGER\_TIMER\_PERIOD = 0x0527 | TY\_FEATURE\_STRUCT, TY\_BOOL\_KEEP\_ALIVE\_ONOFF = 0x0203 | TY\_FEATURE\_BOOL, TY\_INT\_KEEP\_ALIVE\_TIMEOUT = 0x0204 | TY\_FEATURE\_INT, TY\_BOOL\_CMOS\_SYNC = 0x0205 | TY\_FEATURE\_BOOL, TY\_INT\_TRIGGER\_DELAY\_US = 0x0206 | TY\_FEATURE\_INT, TY\_BOOL\_TRIGGER\_OUT\_IO = 0x0207 | TY\_FEATURE\_BOOL, TY\_INT\_TRIGGER\_DURATION\_US = 0x0208 | TY\_FEATURE\_INT, TY ENUM STREAM ASYNC = 0x0209 | TY FEATURE ENUM, TY INT CAPTURE TIME US = 0x0210 | TY FEATURE INT, TY ENUM TIME SYNC TYPE = 0x0211 | TY FEATURE ENUM, TY BOOL TIME SYNC READY = 0x0212 | TY FEATURE BOOL, TY BOOL IR FLASHLIGHT = 0x0213 | TY FEATURE BOOL, TY\_INT\_IR\_FLASHLIGHT\_INTENSITY = 0x0214 | TY\_FEATURE\_INT, TY\_STRUCT\_DO0\_WORKMODE = 0x0215 | TY FEATURE STRUCT, TY STRUCT DI0 WORKMODE = 0x0216 | TY FEATURE STRUCT, TY STRUCT DO1 WORKMODE = 0x0217 | TY FEATURE STRUCT, TY\_STRUCT\_DI1\_WORKMODE = 0x0218 | TY\_FEATURE\_STRUCT, TY\_STRUCT\_DO2\_WORKMODE = 0x0219 | TY FEATURE STRUCT, TY STRUCT DI2 WORKMODE = 0x0220 | TY FEATURE STRUCT, TY BOOL RGB FLASHLIGHT = 0x0221 | TY FEATURE BOOL, TY\_INT\_RGB\_FLASHLIGHT\_INTENSITY = 0x0222 | TY\_FEATURE INT, TY\_ENUM\_CONFIG MODE = 0x0221 | TY FEATURE ENUM, TY ENUM TEMPERATURE ID = 0x0223 | TY FEATURE ENUM, TY ↔ **STRUCT\_TEMPERATURE** = 0x0224 | TY FEATURE STRUCT, TY BOOL AUTO EXPOSURE = 0x0300 | TY FEATURE BOOL, TY INT EXPOSURE TIME = 0x0301 TY FEATURE INT, TY BOOL AUTO GAIN = 0x0302 | TY FEATURE BOOL, TY INT GAIN = 0x0303 | TY FEATURE\_INT, TY\_BOOL\_AUTO\_AWB = 0x0304 | TY\_FEATURE\_BOOL, TY\_STRUCT\_AEC\_ROI = 0x0305 | TY\_FEAT ↔ URE\_STRUCT, TY\_INT\_TOF\_HDR\_RATIO = 0x0306 | TY\_FEATURE\_INT, TY\_INT\_TOF\_JITTER\_THRESHOLD = 0x0307 | TY\_FEATURE\_INT, TY\_FLOAT\_EXPOSURE\_TIME\_US = 0x0308 | TY\_FEATURE\_FLOAT, TY\_INT\_LASER\_POWER = 0x0500 TY FEATURE INT, TY BOOL LASER AUTO CTRL = 0x0501 | TY FEATURE BOOL, TY STRUCT ← LASER PATTERN = 0x0502 | TY FEATURE STRUCT. TY\_INT\_LASER\_CAM\_TRIG\_POS = 0x0503 | TY\_FEATURE\_INT, TY\_INT\_LASER\_CAM\_TRIG\_LEN = 0x0504 | TY FEATURE INT, TY INT LASER LUT TRIG POS = 0x0505 | TY FEATURE INT, TY INT ← **LASER LUT NUM** = 0x0506 | TY FEATURE INT, TY INT LASER PATTERN OFFSET = 0x0507 | TY FEATURE INT, TY INT LASER MIRROR NUM = 0x0508 | TY FEATURE INT, TY INT LASER MIRROR SEL = 0x0509 | TY FEATURE INT, TY INT L↔

```
TY INT LASER FACET IDX = 0x050b | TY FEATURE INT, TY INT LASER FACET POS = 0x050c |
 TY FEATURE INT, TY INT LASER MODE = 0x050d | TY FEATURE INT, TY INT CONST DRV DUTY
 = 0x050e | TY FEATURE INT,
 TY_STRUCT_LASER_ENABLE_BY_IDX = 0x0530 | TY_FEATURE_STRUCT, TY_STRUCT_LASER_POWER_BY_IDX
 = 0x0531 | TY FEATURE STRUCT, TY STRUCT FLOOD ENABLE BY IDX = 0x0532 | TY FEATURE ↔
 STRUCT, TY STRUCT FLOOD POWER BY IDX = 0x0533 | TY FEATURE STRUCT,
 TY BOOL UNDISTORTION = 0x0510 | TY FEATURE BOOL, TY BOOL BRIGHTNESS HISTOGRAM
 = 0x0511 | TY FEATURE BOOL, TY BOOL DEPTH POSTPROC = 0x0512 | TY FEATURE BOOL,
 TY INT R GAIN = 0x0520 | TY FEATURE INT,
 TY INT G GAIN = 0x0521 | TY FEATURE INT, TY INT B GAIN = 0x0522 | TY FEATURE INT,
 TY INT ANALOG GAIN = 0x0524 | TY FEATURE INT, TY BOOL HDR = 0x0525 | TY FEATURE ↔
 BOOL,
 TY_BYTEARRAY_HDR_PARAMETER = 0x0526 | TY_FEATURE_BYTEARRAY, TY_INT_AE_TARGE ↔
 T Y = 0x0527 | TY FEATURE INT, TY BOOL IMU DATA ONOFF = 0x0600 | TY FEATURE BOOL,
 TY_STRUCT_IMU_ACC_BIAS = 0x0601 | TY_FEATURE_STRUCT,
 TY_STRUCT_IMU_ACC_MISALIGNMENT = 0x0602 | TY_FEATURE_STRUCT, TY_STRUCT_IMU_ACC_SCALE
 = 0x0603 | TY FEATURE STRUCT, TY STRUCT IMU GYRO BIAS = 0x0604 | TY FEATURE STRUCT,
 TY STRUCT IMU GYRO MISALIGNMENT = 0x0605 | TY FEATURE STRUCT,
 TY STRUCT IMU GYRO SCALE = 0x0606 | TY FEATURE STRUCT, TY STRUCT IMU CAM TO IMU
 = 0x0607 | TY FEATURE STRUCT, TY ENUM IMU FPS = 0x0608 | TY FEATURE ENUM, TY INT SGBM IMAGE NUM
 = 0x0610 | TY FEATURE INT,
 TY INT SGBM DISPARITY NUM = 0x0611 | TY FEATURE INT, TY INT SGBM DISPARITY OFFSET
 = 0x0612 | TY_FEATURE_INT, TY_INT_SGBM_MATCH_WIN_HEIGHT = 0x0613 | TY_FEATURE_INT,
 TY INT SGBM SEMI PARAM P1 = 0x0614 | TY FEATURE INT,
 TY INT SGBM SEMI PARAM P2 = 0x0615 | TY FEATURE INT, TY INT SGBM UNIQUE FACTOR
 = 0x0616 | TY_FEATURE_INT, TY_INT_SGBM_UNIQUE_ABSDIFF = 0x0617 | TY_FEATURE_INT,
 TY INT SGBM UNIQUE MAX_COST = 0x0618 | TY_FEATURE_INT,
 TY BOOL SGBM HFILTER HALF WIN = 0x0619 | TY_FEATURE_BOOL, TY_INT_SGBM_MATCH_WIN_WIDTH
 = 0x061A | TY FEATURE INT, TY BOOL SGBM MEDFILTER = 0x061B | TY FEATURE BOOL,
 TY BOOL SGBM LRC = 0x061C | TY FEATURE BOOL,
 TY_INT_SGBM_LRC_DIFF = 0x061D | TY_FEATURE_INT, TY_INT_SGBM_MEDFILTER_THRESH =
 0x061E | TY_FEATURE_INT, TY_INT_SGBM_SEMI_PARAM_P1_SCALE = 0x061F | TY_FEATURE INT,
 TY INT SGPM PHASE NUM = 0x0620 | TY FEATURE INT,
 TY_INT_SGPM_NORMAL_PHASE_SCALE = 0x0621 | TY_FEATURE_INT, TY_INT_SGPM_NORMAL_PHASE_OFFSET
 = 0x0622 | TY_FEATURE_INT, TY_INT_SGPM_REF_PHASE_SCALE = 0x0623 | TY_FEATURE_INT,
 TY_INT_SGPM_REF_PHASE_OFFSET = 0x0624 | TY_FEATURE_INT,
 TY FLOAT SGPM EPI HS = 0x0625 | TY FEATURE FLOAT, TY INT SGPM EPI HF = 0x0626 | TY \leftrightarrow
 FEATURE_INT, TY_BOOL_SGPM_EPI_EN = 0x0627 | TY_FEATURE_BOOL, TY_INT_SGPM_EPI_CH0 =
 0x0628 | TY FEATURE INT,
 TY INT SGPM EPI CH1 = 0x0629 | TY FEATURE INT, TY INT SGPM EPI THRESH = 0x062A
 TY_FEATURE_INT, TY_BOOL_SGPM_ORDER_FILTER_EN = 0x062B | TY_FEATURE_BOOL,
 TY INT SGPM ORDER FILTER CHN = 0x062C | TY FEATURE INT,
 TY INT DEPTH MIN MM = 0x062D | TY FEATURE INT, TY INT DEPTH MAX MM = 0x062E | TY FE ↔
 ATURE INT, TY INT SGBM TEXTURE OFFSET = 0x062F | TY FEATURE INT, TY INT SGBM TEXTURE THRESH
 = 0x0630 | TY FEATURE INT.
 TY_STRUCT_PHC_GROUP_ATTR = 0x0710 | TY_FEATURE_STRUCT, TY_ENUM_DEPTH_QUALITY
 = 0x0900 | TY_FEATURE_ENUM, TY_INT_FILTER_THRESHOLD = 0x0901 | TY_FEATURE_INT,
 TY INT TOF CHANNEL = 0x0902 | TY FEATURE INT,
 TY_INT_TOF_MODULATION_THRESHOLD = 0x0903 | TY_FEATURE_INT, TY_STRUCT_TOF_FREQ =
 0x0904 | TY_FEATURE_STRUCT, TY_BOOL_TOF_ANTI_INTERFERENCE = 0x0905 | TY_FEATURE_←
 BOOL, TY INT TOF ANTI SUNLIGHT INDEX = 0x0906 | TY FEATURE INT,
 TY INT MAX SPECKLE SIZE = 0x0907 | TY FEATURE INT, TY INT MAX SPECKLE DIFF = 0x0908 |
 TY FEATURE INT }
    feature for component definitions

    enum TY CONFIG MODE LIST: uint32 t {
```

TY CONFIG MODE PRESETO = 0. TY CONFIG MODE PRESET1, TY CONFIG MODE PRESET2.

Generated by Doxygen

TY CONFIG MODE USERSET0 = (1 << 16),

TY\_CONFIG\_MODE\_USERSET1, TY\_CONFIG\_MODE\_USERSET2 }

```
• enum TY_DEPTH_QUALITY_LIST : uint32_t { TY_DEPTH_QUALITY_BASIC = 1, TY_DEPTH_QUALIT \leftarrow
    Y MEDIUM = 2, TY DEPTH QUALITY HIGH = 4 }

    enum TY TRIGGER POL LIST: uint32 t { TY TRIGGER POL FALLINGEDGE = 0, TY TRIGGER P←

    OL_RISINGEDGE = 1 }
           set external trigger signal edge
enum TY INTERFACE_TYPE_LIST : uint32_t {
    TY_INTERFACE_UNKNOWN = 0, TY_INTERFACE_RAW = 1, TY_INTERFACE_USB = 2, TY_INTERF←
    ACE ETHERNET = 4,
    TY INTERFACE IEEE80211 = 8, TY INTERFACE ALL = 0xffff }

    enum TY ACCESS MODE LIST: uint32 t { TY ACCESS READABLE = 0x1, TY ACCESS WRITABLE

    = 0x2

    enum TY STREAM ASYNC MODE LIST: uint32 t {

    TY STREAM ASYNC OFF = 0, TY STREAM ASYNC DEPTH = 1, TY STREAM ASYNC RGB = 2, T←
    Y_STREAM_ASYNC_DEPTH_RGB = 3,
    TY_STREAM_ASYNC_ALL = 0xff }
          stream async mode

    enum TY PIXEL BITS LIST: uint32 t {

    TY_PIXEL_8BIT = 0x1 << 28, TY_PIXEL_16BIT = 0x2 << 28, TY_PIXEL_24BIT = 0x3 << 28, TY_PIX\leftarrow
    EL 32BIT = 0x4 << 28,
    TY_PIXEL_10BIT = 0x5 << 28, TY_PIXEL_12BIT = 0x6 << 28, TY_PIXEL_14BIT = 0x7 << 28, TY_PI\leftrightarrow
    XEL 48BIT = (uint32 t)0x8 << 28,
    TY_PIXEL_64BIT = (uint32_t)0xa << 28 }
enum TY PIXEL FORMAT LIST: uint32 t {
    TY PIXEL FORMAT UNDEFINED = 0, TY PIXEL FORMAT MONO = (TY PIXEL 8BIT | (0x0 << 24)),
    TY PIXEL FORMAT BAYER8GB = (TY PIXEL 8BIT | (0x1 << 24)), TY PIXEL FORMAT BAYER8BG =
    (TY PIXEL 8BIT | (0x2 << 24)),
    TY PIXEL FORMAT BAYER8GR = (TY PIXEL 8BIT | (0x3 << 24)), TY PIXEL FORMAT BAYER8RG
    = (TY PIXEL 8BIT | (0x4 << 24)), TY PIXEL FORMAT BAYER8GRBG = TY PIXEL FORMAT BAY↔
    ER8GB, TY PIXEL FORMAT BAYER8RGGB = TY PIXEL FORMAT BAYER8BG,
    TY PIXEL FORMAT BAYER8GBRG = TY PIXEL FORMAT BAYER8GR, TY PIXEL FORMAT BAY↔
    ER8BGGR = TY PIXEL FORMAT BAYER8RG, TY PIXEL FORMAT CSI MONO10 = (TY PIXEL 10BIT
    \mid (0x0 << 24)), TY_PIXEL_FORMAT_CSI_BAYER10GRBG = (TY_PIXEL_10BIT \mid (0x1 << 24)),
    TY_PIXEL_FORMAT_CSI_BAYER10RGGB = (TY_PIXEL_10BIT | (0x2 << 24)), TY_PIXEL_FORMAT_CSI_BAYER10GBRG
    = (TY PIXEL 10BIT | (0x3 << 24)), TY PIXEL FORMAT CSI BAYER10BGGR = (TY PIXEL 10BIT | (0x4
    << 24)), TY PIXEL FORMAT CSI MONO12 = (TY PIXEL 12BIT | (0x0 << 24)),
    TY_PIXEL_FORMAT_CSI_BAYER12GRBG = (TY_PIXEL_12BIT | (0x1 << 24)), TY_PIXEL_FORMAT_CSI_BAYER12RGGB
    = (TY PIXEL 12BIT | (0x2 << 24)), TY PIXEL FORMAT CSI BAYER12GBRG = (TY PIXEL 12BIT | (0x3
    << 24)), TY PIXEL FORMAT CSI BAYER12BGGR = (TY PIXEL 12BIT | (0x4 << 24)),
    TY_PIXEL_FORMAT_DEPTH16 = (TY_PIXEL_16BIT | (0x0 << 24)), TY_PIXEL_FORMAT YVYU =
    (TY_PIXEL_16BIT \mid (0x1 << 24)), TY_PIXEL_FORMAT_YUYV = (TY_PIXEL_16BIT \mid (0x2 << 24)),
    TY_PIXEL_FORMAT_MONO16 = (TY_PIXEL_16BIT | (0x3 << 24)),
    TY PIXEL FORMAT TOF IR MONO16 = (TY PIXEL 64BIT | (0x4 << 24)), TY PIXEL FORMAT RGB
    = (TY_PIXEL_24BIT | (0x0 << 24)), TY_PIXEL_FORMAT_BGR = (TY_PIXEL_24BIT | (0x1 << 24)),
    TY PIXEL FORMAT JPEG = (TY PIXEL 24BIT | (0x2 << 24)),
    TY PIXEL FORMAT MJPG = (TY PIXEL 24BIT | (0x3 << 24)), TY PIXEL FORMAT RGB48 = (T↔
    Y PIXEL 48BIT | (0x0 << 24)), TY PIXEL FORMAT BGR48 = (TY PIXEL 48BIT | (0x1 << 24)),
    TY PIXEL FORMAT XYZ48 = (TY PIXEL 48BIT \mid (0x2 << 24)) }
          pixel format definitions
• enum TY_RESOLUTION_MODE_LIST : uint32_t {
    TY RESOLUTION MODE 160x100 = (160 << 12) + 100, TY RESOLUTION MODE 160x120 = (160 << 12) + 120,
    TY_RESOLUTION_MODE_240x320 = (240 << 12) + 320, TY_RESOLUTION_MODE_320x180 = (320 << 12) + 180,
    TY_RESOLUTION_MODE_320x200 = (320 << 12) + 200, TY_RESOLUTION_MODE_320x240 = (320 << 12) + 240,
    TY_RESOLUTION_MODE_480x640 = (480 << 12) +640, TY_RESOLUTION_MODE_640x360 = (640 << 12) +360, TY_RESOLUTION_MODE_640x
    TY RESOLUTION MODE 640x400 = (640 << 12) +400, TY RESOLUTION MODE 640x480 = (640 << 12) +480,
    TY RESOLUTION MODE 960 \times 1280 = (960 < <12) + 1280, TY RESOLUTION MODE 1280 \times 720 = 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 \times 1280 
    (1280 << 12) + 720,
    TY RESOLUTION MODE 1280x800 = (1280<<12)+800, TY RESOLUTION MODE 1280x960
```

```
(1280<<12)+960, TY_RESOLUTION_MODE_1600x1200 = (1600<<12)+1200, TY_RESOLUTION_MODE_800x600
 = (800 << 12) +600
 TY RESOLUTION MODE 1920x1080 = (1920<<12)+1080, TY RESOLUTION MODE 2560x1920 =
 (2560 < < 12) + 1920, TY_RESOLUTION_MODE_2592x1944 = (2592 < < 12) + 1944, TY_RESOLUTION_MODE_1920x1440
 = (1920 << 12) + 1440,
 TY RESOLUTION MODE 240x96 = (240<<12)+96, TY RESOLUTION MODE 2048x1536 = (2048<<12)+1536
    predefined resolution list
enum TY_IMAGE_MODE_LIST : uint32_t {
 TY DECLARE IMAGE MODE1 =(MONO), TY DECLARE IMAGE MODE1 =(MONO), TY DECLARE .
 IMAGE MODE1 = (MONO), TY_DECLARE IMAGE MODE1 = (MONO),
 TY DECLARE IMAGE MODE1 =(MONO), TY DECLARE IMAGE MODE1 =(MONO), TY DECLARE -
 IMAGE_MODE1 =(MONO), TY_DECLARE_IMAGE_MODE1 =(MONO),
 TY DECLARE IMAGE MODE1 =(MONO), TY DECLARE IMAGE MODE1 =(MONO), TY DECLARE -
 IMAGE MODE1 = (MONO), TY DECLARE IMAGE MODE1 = (MONO),
 TY DECLARE IMAGE MODE1 =(MONO), TY DECLARE IMAGE MODE1 =(MONO), TY DECLARE -
 IMAGE_MODE1 =(MONO), TY_DECLARE_IMAGE_MODE1 =(MONO),
 TY DECLARE IMAGE MODE1 = (MONO), TY DECLARE IMAGE MODE1 = (MONO), TY DECLARE -
 IMAGE MODE1 = (MONO), TY DECLARE IMAGE MODE1 = (MONO),
 TY_DECLARE_IMAGE_MODE1 =(MONO), TY_DECLARE_IMAGE_MODE1 =(MONO), TY_DECLARE_←
 IMAGE_MODE1 =(MONO), TY_DECLARE_IMAGE_MODE1 =(MONO),
 TY_DECLARE_IMAGE_MODE1 =(MONO), TY_DECLARE_IMAGE_MODE1 =(MONO), TY_DECLARE_←
 IMAGE_MODE1 = (MONO), TY_DECLARE IMAGE_MODE1 = (MONO),
 TY DECLARE IMAGE MODE1 = (MONO), TY DECLARE IMAGE MODE1 = (MONO) }
    Predefined Image Mode List image mode controls image resolution & format predefined image modes named like
    TY_IMAGE_MODE_MONO_160x120,TY_IMAGE_MODE_RGB_1280x960.
enum TY_TRIGGER_MODE_LIST : uint32_t {
 TY_TRIGGER_MODE_OFF = 0, TY_TRIGGER_MODE_SLAVE = 1, TY_TRIGGER_MODE_M_SIG = 2,
 TY_TRIGGER_MODE_M_PER = 3,
 TY_TRIGGER_MODE_SIG_PASS = 18, TY_TRIGGER_MODE_PER_PASS = 19, TY_TRIGGER_MODE ←
 TIMER LIST = 20, TY TRIGGER MODE TIMER PERIOD = 21,
 TY TRIGGER MODE28 = 28, TY TRIGGER MODE29 = 29, TY TRIGGER MODE PER PASS2 = 30,
 TY TRIGGER WORK MODE31 = 31,
 TY_TRIGGER_MODE_SIG_LASER = 34 }

    enum TY TIME SYNC TYPE LIST: uint32 t {

 TY_TIME_SYNC_TYPE_NONE = 0, TY_TIME_SYNC_TYPE_HOST = 1, TY_TIME_SYNC_TYPE_NTP = 2,
 TY\_TIME\_SYNC\_TYPE\_PTP = 3,
 TY_TIME_SYNC_TYPE_CAN = 4, TY_TIME_SYNC_TYPE_PTP_MASTER = 5 }
    type of time sync

    enum TY LOG LEVEL LIST {

 TY LOG LEVEL VERBOSE = 1, TY LOG LEVEL DEBUG = 2, TY LOG LEVEL INFO = 3, TY LOG ←
  LEVEL WARNING = 4,
 TY_LOG_LEVEL_ERROR = 5, TY_LOG_LEVEL_NEVER = 9 }
• enum TY_VISIBILITY_TYPE { BEGINNER = 0, EXPERT = 1, GURU = 2 }

    enum { TY PATTERN SINE TYPE = 0, TY PATTERN GRAY TYPE, TY PATTERN BIN TYPE, TY ←

 PATTERN EMPTY TYPE = 0xffffffff }

    enum { TY NORMAL PHASE TYPE = 0, TY REFER PHASE TYPE }

enum TY IMU FPS LIST { TY IMU FPS 100HZ = 0, TY IMU FPS 200HZ, TY IMU FPS 400HZ }
```

#### **Variables**

- typedef enum
- typedef **TY\_DO\_5V** = 1
- typedef **TY DO 12V** = 2
- typedef TY\_E\_VOLT\_T\_LIST

```
typedef TY_DO_HIGH = 1
```

- typedef TY\_DO\_PWM = 2
- typedef TY\_DO\_CAM\_TRIG = 3
- typedef TY\_E\_DO\_MODE\_LIST
- typedef TY DI NE INT = 1
- typedef TY DI PE INT = 2
- typedef TY\_E\_DI\_MODE\_LIST
- typedef TY\_DI\_INT\_TRIG\_CAP = 1
- typedef TY\_DI\_INT\_EVENT = 2
- typedef TY\_E\_DI\_INT\_ACTION\_LIST
- typedef TY\_TEMPERATURE\_RIGHT = 1
- typedef TY\_TEMPERATURE\_COLOR = 2
- typedef TY\_TEMPERATURE\_CPU = 3
- typedef TY\_TEMPERATURE\_MAIN\_BOARD = 4
- typedef TY\_TEMPERATURE\_ID\_LIST

#### 5.3.1 Detailed Description

TYDefs.h includes camera control and data receiving data definitions which supports configuration for image resolution, frame rate, exposure time, gain, working mode,etc.

#### 5.3.2 Macro Definition Documentation

## 5.3.2.1 TY\_DECLARE\_IMAGE\_MODE1

```
#define TY_DECLARE_IMAGE_MODE1( pix )
```

#### Value:

```
TY_DECLARE_IMAGE_MODE0(pix, 160x100),
TY_DECLARE_IMAGE_MODE0(pix, 160x120),
TY_DECLARE_IMAGE_MODE0(pix, 320x180),
TY_DECLARE_IMAGE_MODE0(pix, 320x200),
TY_DECLARE_IMAGE_MODE0(pix, 320x240),
TY_DECLARE_IMAGE_MODE0(pix, 480x640),
TY_DECLARE_IMAGE_MODEO(pix, 640x360),
TY_DECLARE_IMAGE_MODEO(pix, 640x400),
TY_DECLARE_IMAGE_MODE0(pix, 640x480),
TY_DECLARE_IMAGE_MODE0(pix, 960x1280),
TY_DECLARE_IMAGE_MODE0(pix, 1280x720),
TY_DECLARE_IMAGE_MODE0(pix, 1280x960),
TY_DECLARE_IMAGE_MODE0(pix, 1280x800),
TY_DECLARE_IMAGE_MODE0(pix, 1600x1200),
TY_DECLARE_IMAGE_MODE0(pix, 800x600), \
TY_DECLARE_IMAGE_MODE0(pix, 1920x1080),
TY_DECLARE_IMAGE_MODE0(pix, 2560x1920),
TY_DECLARE_IMAGE_MODE0(pix, 2592x1944),
TY_DECLARE_IMAGE_MODEO(pix, 1920x1440),
TY_DECLARE_IMAGE_MODEO(pix, 2048x1536),
TY_DECLARE_IMAGE_MODE0 (pix, 240x96)
```

Definition at line 541 of file TYDefs.h.

## 5.3.3 Typedef Documentation

## 5.3.3.1 TY\_ACC\_BIAS

typedef struct TY\_ACC\_BIAS TY\_ACC\_BIAS

a 3x3 matrix

•		
BIASx	BIASy	BIASz

## 5.3.3.2 TY\_ACC\_MISALIGNMENT

typedef struct TY\_ACC\_MISALIGNMENT TY\_ACC\_MISALIGNMENT

#### a 3x3 matrix

•	•	
1	-GAMAyz	GAMAzy
GAMAxz	1	-GAMAzx
-GAMAxy	GAMAyx	1

## 5.3.3.3 TY\_ACC\_SCALE

typedef struct TY\_ACC\_SCALE TY\_ACC\_SCALE

a 3x3 matrix

	•	•
SCALEx	0	0
0	SCALEy	0
0	0	SCALEz

## 5.3.3.4 TY\_ACCESS\_MODE\_LIST

typedef enum TY\_ACCESS\_MODE\_LIST TY\_ACCESS\_MODE\_LIST

Indicate a feature is readable or writable

See also

**TYGetFeatureInfo** 

## 5.3.3.5 TY\_BYTEARRAY\_ATTR

```
typedef struct TY_BYTEARRAY_ATTR TY_BYTEARRAY_ATTR
```

byte array data structure

See also

**TYGetByteArray** 

## 5.3.3.6 TY\_CAMERA\_CALIB\_INFO

typedef struct TY\_CAMERA\_CALIB\_INFO TY\_CAMERA\_CALIB\_INFO

camera 's cailbration data

See also

**TYGetStruct** 

#### 5.3.3.7 TY\_CAMERA\_DISTORTION

```
typedef struct TY_CAMERA_DISTORTION TY_CAMERA_DISTORTION
```

camera distortion parameters

See also

```
TYGetStruct Usage:
```

```
TY_CAMERA_DISTORTION distortion;
TYGetStruct(hDevice, some_compoent, TY_STRUCT_CAM_DISTORTION, &distortion, sizeof(distortion));
```

## 5.3.3.8 TY\_CAMERA\_EXTRINSIC

```
typedef struct TY_CAMERA_EXTRINSIC TY_CAMERA_EXTRINSIC
```

a 4x4 matrix

r11	r12	r13	t1
r21	r22	r23	t2
r31	r32	r33	t3
0	0	0	1

#### See also

## TYGetStruct Usage:

```
TY_CAMERA_EXTRINSIC extrinsic;
TYGetStruct(hDevice, some_compoent, TY_STRUCT_EXTRINSIC, &extrinsic, sizeof(extrinsic));
```

## 5.3.3.9 TY\_CAMERA\_INTRINSIC

```
typedef struct TY_CAMERA_INTRINSIC TY_CAMERA_INTRINSIC
```

#### a 3x3 matrix

		•
fx	0	СХ
0	fy	су
0	0	1

#### See also

## TYGetStruct Usage:

```
TY_CAMERA_INTRINSIC intrinsic;
TYGetStruct(hDevice, some_compoent, TY_STRUCT_CAM_INTRINSIC, &intrinsic, sizeof(intrinsic));
```

## 5.3.3.10 TY\_CAMERA\_ROTATION

```
typedef struct TY_CAMERA_ROTATION TY_CAMERA_ROTATION
```

## a 3x3 matrix

•		
r00	r01	r02
r10	r11	r12
r20	r21	r22

#### See also

#### TYGetStruct Usage:

```
TY_CAMERA_ROTATION rotation;
TYGetStruct(hDevice, some_compoent, TY_STRUCT_CAM_ROTATION, &rotation, sizeof(rotation));
```

## 5.3.3.11 TY\_CAMERA\_TO\_IMU

typedef struct TY\_CAMERA\_TO\_IMU TY\_CAMERA\_TO\_IMU

a 4x4 matrix

r11	r12	r13	t1
r21	r22	r23	t2
r31	r32	r33	t3
0	0	0	1

## 5.3.3.12 TY\_COMPONENT\_ID

typedef uint32\_t TY\_COMPONENT\_ID

component unique id

See also

TY\_DEVICE\_COMPONENT\_LIST

Definition at line 190 of file TYDefs.h.

## 5.3.3.13 TY\_DEVICE\_BASE\_INFO

typedef struct TY\_DEVICE\_BASE\_INFO TY\_DEVICE\_BASE\_INFO

See also

**TYGetDeviceList** 

## 5.3.3.14 TY\_DEVICE\_COMPONENT\_LIST

typedef enum TY\_DEVICE\_COMPONENT\_LIST TY\_DEVICE\_COMPONENT\_LIST

@breif Device Component list A device contains several component. Each component can be controlled by its own features, such as image width, exposure time, etc..

See also

To Know how to get feature information please refer to sample code DumpAllFeatures

5.3 TYDefs.h File Reference 123

## 5.3.3.15 TY\_ENUM\_ENTRY

typedef struct TY\_ENUM\_ENTRY TY\_ENUM\_ENTRY

enum feature entry information

See also

TYGetEnumEntryInfo

## 5.3.3.16 TY\_FEATURE\_ID

typedef uint32\_t TY\_FEATURE\_ID

feature unique id

See also

TY\_FEATURE\_ID\_LIST

Definition at line 373 of file TYDefs.h.

## 5.3.3.17 TY\_FLOAT\_RANGE

typedef struct TY\_FLOAT\_RANGE TY\_FLOAT\_RANGE

float range data structure

See also

TYGetFloatRange

## 5.3.3.18 TY\_GYRO\_BIAS

typedef struct TY\_GYRO\_BIAS TY\_GYRO\_BIAS

a 3x3 matrix



## 5.3.3.19 TY\_GYRO\_MISALIGNMENT

typedef struct TY\_GYRO\_MISALIGNMENT TY\_GYRO\_MISALIGNMENT

a 3x3 matrix

1	-ALPHAyz	ALPHAzy
0	1	-ALPHAzx
0	0	1

## 5.3.3.20 TY\_GYRO\_SCALE

typedef struct TY\_GYRO\_SCALE TY\_GYRO\_SCALE

a 3x3 matrix

		•
SCALEx	0	0
0	SCALEy	0
0	0	SCALEz

## 5.3.3.21 TY\_INTERFACE\_INFO

typedef struct TY\_INTERFACE\_INFO TY\_INTERFACE\_INFO

See also

TYGetInterfaceList

## 5.3.3.22 TY\_INTERFACE\_TYPE\_LIST

typedef enum TY\_INTERFACE\_TYPE\_LIST TY\_INTERFACE\_TYPE\_LIST

Interface type definition

See also

**TYGetInterfaceList** 

5.3 TYDefs.h File Reference 125

## 5.3.3.23 TY\_PIXEL\_BITS\_LIST

```
typedef enum TY_PIXEL_BITS_LIST TY_PIXEL_BITS_LIST
```

Pixel size type definitions to define the pixel size in bits

See also

TY\_PIXEL\_FORMAT\_LIST

## 5.3.3.24 TY\_TRIGGER\_MODE\_LIST

```
typedef enum TY_TRIGGER_MODE_LIST TY_TRIGGER_MODE_LIST
```

See also

refer to sample SimpleView\_TriggerMode for detail usage

## 5.3.4 Enumeration Type Documentation

#### 5.3.4.1 TY\_ACCESS\_MODE\_LIST

```
enum TY_ACCESS_MODE_LIST : uint32_t
```

Indicate a feature is readable or writable

See also

**TYGetFeatureInfo** 

Definition at line 427 of file TYDefs.h.

#### 5.3.4.2 TY\_DEVICE\_COMPONENT\_LIST

```
enum TY_DEVICE_COMPONENT_LIST : uint32_t
```

@breif Device Component list A device contains several component. Each component can be controlled by its own features, such as image width, exposure time, etc..

See also

To Know how to get feature information please refer to sample code DumpAllFeatures

## Enumerator

TY_COMPONENT_DEVICE	Abstract component stands for whole device, always enabled.
TY_COMPONENT_DEPTH_CAM	Depth camera.
TY_COMPONENT_IR_CAM_LEFT	Left IR camera.
TY_COMPONENT_IR_CAM_RIGHT	Right IR camera.
TY_COMPONENT_RGB_CAM_LEFT	Left RGB camera.
TY_COMPONENT_RGB_CAM_RIGHT	Right RGB camera.
TY_COMPONENT_LASER	Laser.
TY_COMPONENT_IMU	Inertial Measurement Unit.
TY_COMPONENT_BRIGHT_HISTO	virtual component for brightness histogram of ir
TY_COMPONENT_STORAGE	virtual component for device storage
TY_COMPONENT_RGB_CAM	Some device has only one RGB camera, map it to left.

Definition at line 175 of file TYDefs.h.

## 5.3.4.3 TY\_FEATURE\_ID\_LIST

enum TY\_FEATURE\_ID\_LIST : uint32\_t

feature for component definitions

TY_STRUCT_CAM_INTRINSIC	see TY_CAMERA_INTRINSIC
TY_STRUCT_EXTRINSIC_TO_DEPTH	extrinsic between depth cam and current component, see
	TY_CAMERA_EXTRINSIC
TY_STRUCT_EXTRINSIC_TO_IR_LEFT	extrinsic between left IR and current compoent, see
	TY_CAMERA_EXTRINSIC
TY_STRUCT_CAM_RECTIFIED_ROTATION	see TY_CAMERA_ROTATION
TY_STRUCT_CAM_DISTORTION	see TY_CAMERA_DISTORTION
TY_STRUCT_CAM_CALIB_DATA	see TY_CAMERA_CALIB_INFO
TY_STRUCT_CAM_RECTIFIED_INTRI	the rectified intrinsic. see TY_CAMERA_INTRINSIC
TY_BYTEARRAY_CUSTOM_BLOCK	used for reading/writing custom block
TY_BYTEARRAY_ISP_BLOCK	used for reading/writing fpn block
TY_INT_PACKET_DELAY	microseconds
TY_INT_NTP_SERVER_IP	Ntp server IP.
TY_INT_LINK_CMD_TIMEOUT	milliseconds
TY_STRUCT_CAM_STATISTICS	statistical information, see TY_CAMERA_STATISTICS
TY_INT_WIDTH	Image width.
TY_INT_HEIGHT	Image height.
TY_ENUM_IMAGE_MODE	Resolution-PixelFromat mode, see TY_IMAGE_MODE_LIST.
TY_FLOAT_SCALE_UNIT	scale unit depth image is uint16 pixel format with default
	millimeter unit ,for some device can output Sub-millimeter
	accuracy data the acutal depth (mm)= PixelValue * ScaleUnit
TY_ENUM_TRIGGER_POL	Trigger POL, see TY_TRIGGER_POL_LIST.
TY_INT_FRAME_PER_TRIGGER	Number of frames captured per trigger.

TY_STRUCT_TRIGGER_PARAM	param of trigger, see TY_TRIGGER_PARAM
TY_STRUCT_TRIGGER_PARAM_EX	param of trigger, see TY_TRIGGER_PARAM_EX
TY_STRUCT_TRIGGER_TIMER_LIST	param of trigger mode 20, see TY_TRIGGER_TIMER_LIST
TY_STRUCT_TRIGGER_TIMER_PERIOD	param of trigger mode 21, see
	TY_TRIGGER_TIMER_PERIOD
TY_BOOL_KEEP_ALIVE_ONOFF	Keep Alive switch.
TY_INT_KEEP_ALIVE_TIMEOUT	Keep Alive timeout.
TY_BOOL_CMOS_SYNC	Cmos sync switch.
TY_INT_TRIGGER_DELAY_US	Trigger delay time, in microseconds.
TY_BOOL_TRIGGER_OUT_IO	Trigger out IO.
TY_INT_TRIGGER_DURATION_US	Trigger duration time, in microseconds.
TY_ENUM_STREAM_ASYNC TY_INT_CAPTURE_TIME_US	stream async switch, see TY_STREAM_ASYNC_MODE capture time in multi-ir
TY ENUM TIME SYNC TYPE	-
TY_BOOL_TIME_SYNC_READY	see TY_TIME_SYNC_TYPE time sync done status
TY_BOOL_IR_FLASHLIGHT	Enable switch for floodlight used in ir component.
TY_INT_IR_FLASHLIGHT_INTENSITY	ir component flashlight intensity level
TY STRUCT DO0 WORKMODE	DO 0 workmode, see TY DO WORKMODE.
TY_STRUCT_DI0_WORKMODE	DI_0 workmode, see TY_DI_WORKMODE.
TY_STRUCT_DO1_WORKMODE	DO_1 workmode, see TY_DO_WORKMODE.
TY_STRUCT_DI1_WORKMODE	DI_1 workmode, see TY_DI_WORKMODE.
TY_STRUCT_DO2_WORKMODE	DO_2 workmode, see TY_DO_WORKMODE.
TY_STRUCT_DI2_WORKMODE	DI_2 workmode, see TY_DI_WORKMODE.
TY_BOOL_RGB_FLASHLIGHT	Enable switch for floodlight used in rgb component.
TY_INT_RGB_FLASHLIGHT_INTENSITY	rgb component flashlight intensity level
TY_BOOL_AUTO_EXPOSURE	Auto exposure switch.
TY_INT_EXPOSURE_TIME	Exposure time.
TY_BOOL_AUTO_GAIN	Auto gain switch.
TY_INT_GAIN	Sensor Gain.
TY_BOOL_AUTO_AWB	Auto white balance.
TY_STRUCT_AEC_ROI	region of aec statistics, see TY_AEC_ROI_PARAM
TY_INT_TOF_HDR_RATIO	tof sensor hdr ratio for depth
TY_INT_TOF_JITTER_THRESHOLD	tof jitter threshold for depth
TY_FLOAT_EXPOSURE_TIME_US	the exposure time, unit: us
TY_INT_LASER_POWER	Laser power level.
TY_BOOL_LASER_AUTO_CTRL	Laser auto ctrl.
TY_STRUCT_LASER_ENABLE_BY_IDX	Laser enable by device index.
TY_STRUCT_LASER_POWER_BY_IDX	Laser power by device index.
TY_STRUCT_FLOOD_ENABLE_BY_IDX	Flood enable by device index.
TY_STRUCT_FLOOD_POWER_BY_IDX	Flood power by device index.
TY_BOOL_UNDISTORTION	Output undistorted image.
TY_BOOL_BRIGHTNESS_HISTOGRAM	Output bright histogram.
TY_BOOL_DEPTH_POSTPROC	Do depth image postproc.
TY_INT_R_GAIN	Gain of R channel.
TY_INT_G_GAIN	Gain of R channel
TY_INT_B_GAIN TY_INT_ANALOG_GAIN	Gain of B channel.
I T_INT_ANALUG_GAIN	Analog gain.

TY_BOOL_HDR	HDR func enable/disable.
TY_BYTEARRAY_HDR_PARAMETER	HDR parameters.
TY_BOOL_IMU_DATA_ONOFF	AE target y. IMU Data Onoff
TY_STRUCT_IMU_ACC_BIAS	IMU acc bias matrix, see TY_ACC_BIAS.
TY_STRUCT_IMU_ACC_MISALIGNMENT	IMU acc misalignment matrix, see
	TY_ACC_MISALIGNMENT.
TY_STRUCT_IMU_ACC_SCALE	IMU acc scale matrix, see TY_ACC_SCALE.
TY_STRUCT_IMU_GYRO_BIAS	IMU gyro bias matrix, see TY_GYRO_BIAS.
TY_STRUCT_IMU_GYRO_MISALIGNMENT	IMU gyro misalignment matrix, see TY_GYRO_MISALIGNMENT.
TY_STRUCT_IMU_GYRO_SCALE	IMU gyro scale matrix, see TY_GYRO_SCALE.
TY_STRUCT_IMU_CAM_TO_IMU	IMU camera to imu matrix, see TY_CAMERA_TO_IMU.
TY_ENUM_IMU_FPS	IMU fps, see TY_IMU_FPS_LIST.
TY_INT_SGBM_IMAGE_NUM	SGBM image channel num.
TY_INT_SGBM_DISPARITY_NUM	SGBM disparity num.
TY_INT_SGBM_DISPARITY_OFFSET	SGBM disparity offset.
TY_INT_SGBM_MATCH_WIN_HEIGHT	SGBM match window height.
TY_INT_SGBM_SEMI_PARAM_P1	SGBM semi global param p1.
TY_INT_SGBM_SEMI_PARAM_P2	SGBM semi global param p2.
TY_INT_SGBM_UNIQUE_FACTOR	SGBM uniqueness factor param.
TY_INT_SGBM_UNIQUE_ABSDIFF	SGBM uniqueness min absolute diff.
TY_INT_SGBM_UNIQUE_MAX_COST	SGBM uniqueness max cost param.
TY_BOOL_SGBM_HFILTER_HALF_WIN	SGBM enable half window size.
TY_INT_SGBM_MATCH_WIN_WIDTH	SGBM match window width.
TY_BOOL_SGBM_MEDFILTER	SGBM enable median filter.
TY_BOOL_SGBM_LRC	SGBM enable left right consist check.
TY_INT_SGBM_LRC_DIFF	SGBM max diff.
TY_INT_SGBM_MEDFILTER_THRESH	SGBM median filter thresh.
TY_INT_SGBM_SEMI_PARAM_P1_SCALE	SGBM semi global param p1 scale.
TY_INT_SGPM_PHASE_NUM	Phase num to calc a depth.
TY_INT_SGPM_NORMAL_PHASE_SCALE	phase scale when calc a depth
TY_INT_SGPM_NORMAL_PHASE_OFFSET	Phase offset when calc a depth.
TY_INT_SGPM_REF_PHASE_SCALE	Reference Phase scale when calc a depth.
TY_INT_SGPM_REF_PHASE_OFFSET	Reference Phase offset when calc a depth.
TY_FLOAT_SGPM_EPI_HS	Epipolar Constraint pattern scale.
TY_INT_SGPM_EPI_HF	Epipolar Constraint pattern offset.
TY_BOOL_SGPM_EPI_EN	Epipolar Constraint enable.
TY_INT_SGPM_EPI_CH0	Epipolar Constraint channel0.
TY_INT_SGPM_EPI_CH1	Epipolar Constraint channel1.
TY_INT_SGPM_EPI_THRESH	Epipolar Constraint thresh.
TY_BOOL_SGPM_ORDER_FILTER_EN	Phase order filter enable.
TY_INT_SGPM_ORDER_FILTER_CHN	Phase order filter channel.
TY_INT_DEPTH_MIN_MM	min depth in mm output
TY_INT_DEPTH_MAX_MM	max depth in mm ouput
TY_INT_SGBM_TEXTURE_OFFSET	texture filter value offset
TY_INT_SGBM_TEXTURE_THRESH	texture filter threshold
TY_STRUCT_PHC_GROUP_ATTR	Phase compute group attribute.

#### Enumerator

TY_ENUM_DEPTH_QUALITY	the quality of generated depth, see TY_DEPTH_QUALITY
TY_INT_FILTER_THRESHOLD	the threshold of the noise filter, 0 for disabled
TY_INT_TOF_CHANNEL	the frequency channel of tof
TY_INT_TOF_MODULATION_THRESHOLD	the threshold of the tof modulation
TY_STRUCT_TOF_FREQ	the frequency of tof, see TY_TOF_FREQ
TY_BOOL_TOF_ANTI_INTERFERENCE	cooperation if multi-device used
TY_INT_TOF_ANTI_SUNLIGHT_INDEX	the index of anti-sunlight
TY_INT_MAX_SPECKLE_SIZE	the max size of speckle
TY_INT_MAX_SPECKLE_DIFF	the max diff of speckle

Definition at line 209 of file TYDefs.h.

## 5.3.4.4 TY\_INTERFACE\_TYPE\_LIST

```
enum TY_INTERFACE_TYPE_LIST : uint32_t
```

Interface type definition

See also

TYGetInterfaceList

Definition at line 414 of file TYDefs.h.

## 5.3.4.5 TY\_PIXEL\_BITS\_LIST

```
enum TY_PIXEL_BITS_LIST : uint32_t
```

Pixel size type definitions to define the pixel size in bits

See also

TY\_PIXEL\_FORMAT\_LIST

Definition at line 449 of file TYDefs.h.

## 5.3.4.6 TY\_PIXEL\_FORMAT\_LIST

```
enum TY_PIXEL_FORMAT_LIST : uint32_t
```

pixel format definitions

## Enumerator

TY_PIXEL_FORMAT_MONO	0x10000000
TY_PIXEL_FORMAT_BAYER8GB	0x11000000
TY_PIXEL_FORMAT_BAYER8BG	0x12000000
TY_PIXEL_FORMAT_BAYER8GR	0x13000000
TY_PIXEL_FORMAT_BAYER8RG	0x14000000
TY_PIXEL_FORMAT_CSI_MONO10	0x50000000
TY_PIXEL_FORMAT_CSI_BAYER10GRBG	0x51000000
TY_PIXEL_FORMAT_CSI_BAYER10RGGB	0x52000000
TY_PIXEL_FORMAT_CSI_BAYER10GBRG	0x53000000
TY_PIXEL_FORMAT_CSI_BAYER10BGGR	0x54000000
TY_PIXEL_FORMAT_CSI_MONO12	0x60000000
TY_PIXEL_FORMAT_CSI_BAYER12GRBG	0x61000000
TY_PIXEL_FORMAT_CSI_BAYER12RGGB	0x62000000
TY_PIXEL_FORMAT_CSI_BAYER12GBRG	0x63000000
TY_PIXEL_FORMAT_CSI_BAYER12BGGR	0x64000000
TY_PIXEL_FORMAT_DEPTH16	0x20000000
TY_PIXEL_FORMAT_YVYU	0x21000000, yvyu422
TY_PIXEL_FORMAT_YUYV	0x22000000, yuyv422
TY_PIXEL_FORMAT_MONO16	0x23000000,
TY_PIXEL_FORMAT_TOF_IR_MONO16	0xa4000000,
TY_PIXEL_FORMAT_RGB	0x30000000
TY_PIXEL_FORMAT_BGR	0x31000000
TY_PIXEL_FORMAT_JPEG	0x32000000
TY_PIXEL_FORMAT_MJPG	0x33000000
TY_PIXEL_FORMAT_RGB48	0x80000000
TY_PIXEL_FORMAT_BGR48	0x81000000
TY_PIXEL_FORMAT_XYZ48	0x82000000

Definition at line 467 of file TYDefs.h.

## 5.3.4.7 TY\_RESOLUTION\_MODE\_LIST

enum TY\_RESOLUTION\_MODE\_LIST : uint32\_t

predefined resolution list

TY_RESOLUTION_MODE_160x100	0x000a0078
TY_RESOLUTION_MODE_160x120	0x000a0078
TY_RESOLUTION_MODE_240x320	0x000f0140
TY_RESOLUTION_MODE_320x180	0x001400b4
TY_RESOLUTION_MODE_320x200	0x001400c8
TY_RESOLUTION_MODE_320x240	0x001400f0
TY_RESOLUTION_MODE_480x640	0x001e0280
TY_RESOLUTION_MODE_640x360	0x00280168

#### Enumerator

TY_RESOLUTION_MODE_640x400	0x00280190
TY_RESOLUTION_MODE_640x480	0x002801e0
TY_RESOLUTION_MODE_960x1280	0x003c0500
TY_RESOLUTION_MODE_1280x720	0x005002d0
TY_RESOLUTION_MODE_1280x800	0x00500320
TY_RESOLUTION_MODE_1280x960	0x005003c0
TY_RESOLUTION_MODE_1600x1200	0x006404b0
TY_RESOLUTION_MODE_800x600	0x00320258
TY_RESOLUTION_MODE_1920x1080	0x00780438
TY_RESOLUTION_MODE_2560x1920	0x00a00780
TY_RESOLUTION_MODE_2592x1944	0x00a20798
TY_RESOLUTION_MODE_1920x1440	0x007805a0
TY_RESOLUTION_MODE_240x96	0x000f0060
TY_RESOLUTION_MODE_2048x1536	0x00800600

Definition at line 511 of file TYDefs.h.

## 5.3.4.8 TY\_TRIGGER\_MODE\_LIST

```
enum TY_TRIGGER_MODE_LIST : uint32_t
```

#### See also

refer to sample SimpleView\_TriggerMode for detail usage

#### Enumerator

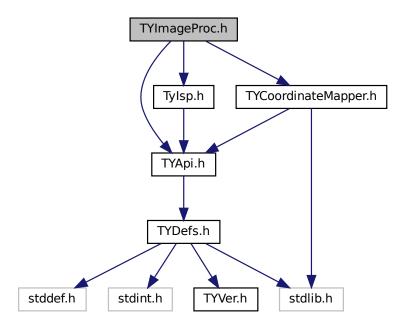
TY_TRIGGER_MODE_OFF	not trigger mode, continuous mode
TY_TRIGGER_MODE_SLAVE	slave mode, receive soft/hardware triggers
TY_TRIGGER_MODE_M_SIG	master mode 1, sending one trigger signal once received a soft trigger
TY_TRIGGER_MODE_M_PER	master mode 2, periodic sending one trigger signals, 'fps' param
	should be set
TY_TRIGGER_MODE_SIG_PASS	discard, using TY_TRIGGER_MODE28
TY_TRIGGER_MODE_PER_PASS	discard, using TY_TRIGGER_MODE29
TY_TRIGGER_MODE_PER_PASS2	trigger mode 30,Alternate output depth image/ir image

Definition at line 612 of file TYDefs.h.

# 5.4 TYImageProc.h File Reference

```
#include "TYApi.h"
#include "TYCoordinateMapper.h"
```

#include "TyIsp.h"
Include dependency graph for TYImageProc.h:



#### **Classes**

- struct DepthSpeckleFilterParameters
- struct DepthEnhenceParameters

#### **Macros**

- #define DepthSpeckleFilterParameters Initializer {150, 64}
- #define DepthEnhenceParameters\_Initializer {10, 20, 10, 0.1f}

#### **Functions**

- TY\_CAPI TYImageProcesAcceEnable (bool en)
  - Image processing acceleration switch.
- TY\_CAPI TYUndistortImage (const TY\_CAMERA\_CALIB\_INFO \*srcCalibInfo, const TY\_IMAGE\_DATA \*srcImage, const TY\_CAMERA\_INTRINSIC \*cameraNewIntrinsic, TY\_IMAGE\_DATA \*dstImage)
  - Do image undistortion, only support  $TY\_PIXEL\_FORMAT\_MONO$ ,  $TY\_PIXEL\_FORMAT\_RGB$ ,  $TY\_PIXEL\_FORMAT\_RGB$ .
- TY\_CAPI TYDepthSpeckleFilter (TY\_IMAGE\_DATA \*depthImage, const DepthSpeckleFilterParameters \*param)
  - Remove speckles on depth image.
- TY\_CAPI TYDepthEnhenceFilter (const TY\_IMAGE\_DATA \*depthImages, int imageNum, TY\_IMAGE\_DATA \*guide, TY\_IMAGE\_DATA \*output, const DepthEnhenceParameters \*param)
  - Remove speckles on depth image.

## 5.4.1 Detailed Description

@breif Image post-process API

## Copyright

Copyright(C)2016-2018 Percipio All Rights Reserved

## 5.4.2 Function Documentation

## 5.4.2.1 TYDepthEnhenceFilter()

Remove speckles on depth image.

#### **Parameters**

in	depthImage	Pointer to depth image array.
in	imageNum	Depth image array size.
in,out	guide	Guide image.
out	output	Output depth image.
in	param	Algorithm parameters.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NULL_POINTER	Any depthImage, param, output or output->buffer is NULL.
TY_STATUS_INVALID_PARAMETER	imageNum >= 11 or imageNum <= 0, or any image invalid
TY_STATUS_OUT_OF_MEMORY	Output image not suitable.

## 5.4.2.2 TYDepthSpeckleFilter()

Remove speckles on depth image.

#### **Parameters**

in,out	depthImage	Depth image to be processed.
in	param	Algorithm parameters.

#### Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NULL_POINTER	Any depth, param or depth->buffer is NULL.
TY_STATUS_INVALID_PARAMETER	param->max_speckle_size <= 0 or param->max_speckle_diff <= 0

## 5.4.2.3 TYImageProcesAcceEnable()

```
TY_CAPI TYImageProcesAcceEnable ( bool en )
```

Image processing acceleration switch.

#### **Parameters**

in	en	Enable image process acceleration switch
----	----	--

## 5.4.2.4 TYUndistortImage()

Do image undistortion, only support TY\_PIXEL\_FORMAT\_MONO , TY\_PIXEL\_FORMAT\_RGB, TY\_PIXEL\_FOR  $\longleftrightarrow$  MAT\_BGR.

## Parameters

in	srcCalibInfo	Image calibration data.
in	srcImage	Source image.
in	cameraNewIntrinsic	Expected new image intrinsic, will use srcCalibInfo for new image intrinsic if set
		to NULL.
out	dstlmage	Output image.

#### Return values

TY_STATUS_OK Succeed.
-----------------------

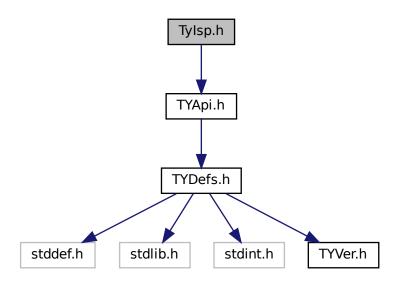
## Return values

TY_STATUS_NULL_POINTER	Any srcCalibInfo, srcImage, dstImage, srcImage->buffer, dstImage->buffer is NULL.
TY_STATUS_INVALID_PARAMETER	Invalid srcImage->width, srcImage->height, dstImage->width, dstImage->height or unsupported pixel format.

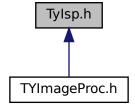
# 5.5 Tylsp.h File Reference

#include "TYApi.h"

Include dependency graph for Tylsp.h:



This graph shows which files directly or indirectly include this file:



#### **Classes**

struct TY\_ISP\_FEATURE\_INFO

#### **Macros**

#define TYISP\_CAPI TY\_CAPI

#### **Typedefs**

typedef void \* TY\_ISP\_HANDLE

#### **Enumerations**

```
enum TY_ISP_FEATURE_ID {
 TY_ISP_FEATURE_CAM_MODEL = 0x000000, TY_ISP_FEATURE_CAM_DEV_HANDLE = 0x000001,
 TY_ISP_FEATURE_CAM_DEV_COMPONENT = 0x000002, TY_ISP_FEATURE_IMAGE_SIZE =
 TY ISP FEATURE WHITEBALANCE GAIN = 0x000200, TY ISP FEATURE ENABLE AUTO WHIT-
 EBALANCE = 0x000300, TY ISP FEATURE SHADING = 0x000400, TY ISP FEATURE SHADING C←
 ENTER = 0 \times 000500,
 TY_ISP_FEATURE_BLACK_LEVEL = 0x000600, TY_ISP_FEATURE_BLACK_LEVEL_COLUMN =
 0x000610, TY_ISP_FEATURE_BLACK_LEVEL_GAIN = 0x000700, TY_ISP_FEATURE_BLACK_LEVEL_GAIN_COLUMN
 = 0x000710,
 TY ISP FEATURE BAYER PATTERN = 0x000800, TY ISP FEATURE DEMOSAIC METHOD =
 0x000900, TY ISP FEATURE GAMMA = 0x0000A00, TY ISP FEATURE DEFECT PIXEL LIST =
 TY ISP FEATURE CCM = 0x000C00, TY ISP FEATURE CCM ENABLE = 0x000C10, TY ISP FEAT ←
 URE BRIGHT = 0x000D00, TY ISP FEATURE CONTRAST = 0x000E00,
 TY ISP FEATURE AUTOBRIGHT = 0x000F00, TY ISP FEATURE INPUT RESAMPLE SCALE =
 0x001000, TY ISP FEATURE ENABLE AUTO EXPOSURE GAIN = 0x001100, TY ISP FEATURE AUTO EXPOSURE I
 = 0x001200.
 TY ISP FEATURE AUTO GAIN RANGE = 0x001300, TY ISP FEATURE AUTO EXPOSURE UPDATE INTERVAL
 = 0x001400, TY_ISP_FEATURE_DEBUG_LOG = 0xff000000 }

    enum TY ISP BAYER PATTERN {

 TY ISP BAYER GB = 0, TY ISP BAYER BG = 1, TY ISP BAYER RG = 2, TY ISP BAYER GR = 3,
 TY_ISP_BAYER_AUTO = 0xff }

    enum TY DEMOSAIC METHOD { TY DEMOSAIC METHOD SIMPLE = 0, TY DEMOSAIC METHOD ←

 BILINEAR = 1, TY DEMOSAIC METHOD HQLINEAR = 2, TY DEMOSAIC METHOD EDGESENSE =
 3 }
```

#### **Functions**

- TYISP CAPI TYISPCreate (TY ISP HANDLE \*handle)
- TYISP\_CAPI TYISPRelease (TY\_ISP\_HANDLE \*handle)
- TYISP\_CAPI TYISPLoadConfig (TY\_ISP\_HANDLE handle, const uint8\_t \*config, uint32\_t config\_size)
- TYISP\_CAPI TYISPUpdateDevice (TY\_ISP\_HANDLE handle)
   @breif called by main thread to update & control device status for ISP
- TYISP\_CAPI TYISPSetFeature (TY\_ISP\_HANDLE handle, TY\_ISP\_FEATURE\_ID feature\_id, const uint8
   t \*data, int32 t size)
- TYISP\_CAPI TYISPGetFeature (TY\_ISP\_HANDLE handle, TY\_ISP\_FEATURE\_ID feature\_id, uint8\_
   t \*data\_buff, int32\_t buff\_size)

- TYISP\_CAPI **TYISPGetFeatureSize** (TY\_ISP\_HANDLE handle, TY\_ISP\_FEATURE\_ID feature\_id, int32\_t \*size)
- TYISP\_CAPI TYISPHasFeature (TY\_ISP\_HANDLE handle, TY\_ISP\_FEATURE\_ID feature\_id)
- TYISP\_CAPI **TYISPGetFeatureInfoList** (TY\_ISP\_HANDLE handle, TY\_ISP\_FEATURE\_INFO \*info\_buffer, int buffer\_size)
- TYISP\_CAPI TYISPGetFeatureInfoListSize (TY\_ISP\_HANDLE handle, int32\_t \*buffer\_size)
- TYISP\_CAPI TYISPProcessImage (TY\_ISP\_HANDLE handle, const TY\_IMAGE\_DATA \*image\_bayer, TY\_IMAGE\_DATA \*image\_out)

@breif convert bayer raw image to rgb image,output buffer is allocated by invoker

## 5.5.1 Detailed Description

@breif this file Include interface declare for raw color image (bayer format) process functions

Copyright(C)2016-2019 Percipio All Rights Reserved

## 5.5.2 Enumeration Type Documentation

## 5.5.2.1 TY\_ISP\_FEATURE\_ID

enum TY\_ISP\_FEATURE\_ID

#### **Enumerator**

TY_ISP_FEATURE_CAM_DEV_HANDLE	device handle for device control
TY_ISP_FEATURE_CAM_DEV_COMPONENT	the component to control
TY_ISP_FEATURE_IMAGE_SIZE	image size width&height
TY_ISP_FEATURE_BLACK_LEVEL	global black level
TY_ISP_FEATURE_BLACK_LEVEL_COLUMN	to set different black level for each image column
TY_ISP_FEATURE_BLACK_LEVEL_GAIN	global pixel gain
TY_ISP_FEATURE_BLACK_LEVEL_GAIN_COLUMN	to set different gain for each image column
TY_ISP_FEATURE_CCM_ENABLE	ENABLE CCM.
TY_ISP_FEATURE_AUTO_EXPOSURE_RANGE	exposure range ,default no limit
TY_ISP_FEATURE_AUTO_GAIN_RANGE	gain range ,default no limit
TY_ISP_FEATURE_AUTO_EXPOSURE_UPDATE_INT↔	update device exposure interval , default 5 frame
ERVAL	
TY_ISP_FEATURE_DEBUG_LOG	display detail log information

Definition at line 17 of file Tylsp.h.

# Index

DepthEnhenceParameters, 7	TY_BOOL_TIME_SYNC_READY
DepthSpeckleFilterParameters, 7	TYDefs.h, 127
	TY_BOOL_TOF_ANTI_INTERFERENCI
pattern_bin_param, 8	TYDefs.h, 129
pattern_gray_param, 8	TY_BOOL_TRIGGER_OUT_IO
pattern_sine_param, 8	TYDefs.h, 127
	TY_BOOL_UNDISTORTION
TY_ACC_BIAS, 9	TYDefs.h, 127
TYDefs.h, 119	TY_BYTEARRAY_ATTR, 11
TY_ACC_MISALIGNMENT, 9	TYDefs.h, 120
TYDefs.h, 119	unit size, 11
TY_ACC_SCALE, 10	valid size, 12
TYDefs.h, 119	TY_BYTEARRAY_CUSTOM_BLOCK
TY_ACCESS_MODE_LIST	TYDefs.h, 126
TYDefs.h, 119, 125	TY_BYTEARRAY_HDR_PARAMETER
TY_AEC_ROI_PARAM, 11	TYDefs.h, 128
TY_BOOL_AUTO_AWB	TY_BYTEARRAY_ISP_BLOCK
TYDefs.h, 127	TYDefs.h, 126
TY_BOOL_AUTO_EXPOSURE	TY_CAMERA_CALIB_INFO, 12
TYDefs.h, 127	TYDefs.h, 120
TY_BOOL_AUTO_GAIN	TY_CAMERA_DISTORTION, 13
TYDefs.h, 127	TYDefs.h, 120
TY_BOOL_BRIGHTNESS_HISTOGRAM	TY_CAMERA_EXTRINSIC, 13
TYDefs.h, 127	TYDefs.h, 120
TY_BOOL_CMOS_SYNC	TY_CAMERA_INTRINSIC, 14
TYDefs.h, 127	TYDefs.h, 121
TY_BOOL_DEPTH_POSTPROC	TY_CAMERA_ROTATION, 15
TYDefs.h, 127	TYDefs.h, 121
TY_BOOL_HDR	TY_CAMERA_STATISTICS, 15
TYDefs.h, 128	TY_CAMERA_TO_IMU, 16
TY_BOOL_IMU_DATA_ONOFF	TYDefs.h, 121
TYDefs.h, 128	TY_COMPONENT_BRIGHT_HISTO
TY_BOOL_IR_FLASHLIGHT	TYDefs.h, 126
TYDefs.h, 127	TY_COMPONENT_DEPTH_CAM
TY_BOOL_KEEP_ALIVE_ONOFF	TYDefs.h, 126
TYDefs.h, 127	TY_COMPONENT_DEVICE
TY_BOOL_LASER_AUTO_CTRL	TYDefs.h, 126
TYDefs.h, 127	TY COMPONENT ID
TY_BOOL_RGB_FLASHLIGHT	TYDefs.h, 122
TYDefs.h, 127	TY COMPONENT IMU
TY_BOOL_SGBM_HFILTER_HALF_WIN	TYDefs.h, 126
TYDefs.h, 128	TY_COMPONENT_IR_CAM_LEFT
TY_BOOL_SGBM_LRC	TYDefs.h, 126
TYDefs.h, 128	
TY_BOOL_SGBM_MEDFILTER	TY_COMPONENT_IR_CAM_RIGHT
TYDefs.h, 128	TYDefs.h, 126
TY_BOOL_SGPM_EPI_EN	TY_COMPONENT_LASER
TYDefs.h, 128	TYDefs.h, 126
TY_BOOL_SGPM_ORDER_FILTER_EN TYDefs.h. 128	TY_COMPONENT_RGB_CAM TYDefs.h. 126
I I DCIS.II. 140	I TUCIS.II. 140

TY_COMPONENT_RGB_CAM_LEFT	TYDefs.h, 127
TYDefs.h, 126	TY_INT_DEPTH_MAX_MM
TY_COMPONENT_RGB_CAM_RIGHT	TYDefs.h, 128
TYDefs.h, 126	TY_INT_DEPTH_MIN_MM
TY_COMPONENT_STORAGE	TYDefs.h, 128
TYDefs.h, 126	TY_INT_EXPOSURE_TIME
TY_DECLARE_IMAGE_MODE1	TYDefs.h, 127
TYDefs.h, 118	TY_INT_FILTER_THRESHOLD
TY_DEVICE_BASE_INFO, 16	TYDefs.h, 129
TYDefs.h, 122	TY_INT_FRAME_PER_TRIGGER
TY_DEVICE_COMPONENT_LIST	TYDefs.h, 126
TYDefs.h, 122, 125	TY INT G GAIN
TY_DEVICE_NET_INFO, 17	TYDefs.h, 127
TY DEVICE USB INFO, 18	TY INT GAIN
TY DI WORKMODE, 18	TYDefs.h, 127
TY_DO_WORKMODE, 19	TY_INT_HEIGHT
TY ENUM DEPTH QUALITY	TYDefs.h, 126
TYDefs.h, 129	TY_INT_IR_FLASHLIGHT_INTENSITY
TY_ENUM_ENTRY, 19	TYDefs.h, 127
TYDefs.h, 122	TY_INT_KEEP_ALIVE_TIMEOUT
TY ENUM IMAGE MODE	TYDefs.h, 127
TYDefs.h, 126	TY_INT_LASER_POWER
,	
TY_ENUM_IMU_FPS	TYDefs.h, 127
TYDefs.h, 128	TY_INT_LINK_CMD_TIMEOUT
TY_ENUM_STREAM_ASYNC	TYDefs.h, 126
TYDefs.h, 127	TY_INT_MAX_SPECKLE_DIFF
TY_ENUM_TIME_SYNC_TYPE	TYDefs.h, 129
TYDefs.h, 127	TY_INT_MAX_SPECKLE_SIZE
TY_ENUM_TRIGGER_POL	TYDefs.h, 129
TYDefs.h, 126	TY_INT_NTP_SERVER_IP
TY_EVENT_INFO, 20	TYDefs.h, 126
TY_FEATURE_ID	TY_INT_PACKET_DELAY
TYDefs.h, 123	TYDefs.h, 126
TY_FEATURE_ID_LIST	TY_INT_R_GAIN
TYDefs.h, 126	TYDefs.h, 127
TY_FEATURE_INFO, 20	TY_INT_RANGE, 25
TY_FLOAT_EXPOSURE_TIME_US	TY_INT_RGB_FLASHLIGHT_INTENSITY
TYDefs.h, 127	TYDefs.h, 127
TY_FLOAT_RANGE, 21	TY_INT_SGBM_DISPARITY_NUM
TYDefs.h, 123	TYDefs.h, 128
TY_FLOAT_SCALE_UNIT	TY_INT_SGBM_DISPARITY_OFFSET
TYDefs.h, 126	TYDefs.h, 128
TY_FLOAT_SGPM_EPI_HS	TY_INT_SGBM_IMAGE_NUM
TYDefs.h, 128	TYDefs.h, 128
TY_FRAME_DATA, 21	TY_INT_SGBM_LRC_DIFF
TY_GYRO_BIAS, 22	TYDefs.h, 128
TYDefs.h, 123	TY INT SGBM MATCH WIN HEIGHT
TY_GYRO_MISALIGNMENT, 22	TYDefs.h, 128
TYDefs.h, 124	TY_INT_SGBM_MATCH_WIN_WIDTH
TY_GYRO_SCALE, 23	TYDefs.h, 128
TYDefs.h, 124	TY_INT_SGBM_MEDFILTER_THRESH
TY_IMAGE_DATA, 24	
	TYDefs.h, 128
TY_IMU_DATA, 24	TY_INT_SGBM_SEMI_PARAM_P1
TY_INT_ANALOG_GAIN	TYDefs.h, 128
TYDefs.h, 127	TY_INT_SGBM_SEMI_PARAM_P1_SCALI
TY_INT_B_GAIN	TYDefs.h, 128
TYDefs.h, 127	TY_INT_SGBM_SEMI_PARAM_P2
TY_INT_CAPTURE_TIME_US	TYDefs.h, 128

TY_INT_SGBM_TEXTURE_OFFSET	TY_ISP_FEATURE_BLACK_LEVEL_COLUMN
TYDefs.h, 128	Tylsp.h, 137
TY_INT_SGBM_TEXTURE_THRESH	TY_ISP_FEATURE_BLACK_LEVEL_GAIN
TYDefs.h, 128	Tylsp.h, 137
TY INT SGBM UNIQUE ABSDIFF	TY_ISP_FEATURE_BLACK_LEVEL_GAIN_COLUMN
TYDefs.h, 128	Tylsp.h, 137
TY_INT_SGBM_UNIQUE_FACTOR	TY_ISP_FEATURE_CAM_DEV_COMPONENT
TYDefs.h, 128	Tylsp.h, 137
TY_INT_SGBM_UNIQUE_MAX_COST	TY_ISP_FEATURE_CAM_DEV_HANDLE
TYDefs.h, 128	Tylsp.h, 137
TY_INT_SGPM_EPI_CH0	TY_ISP_FEATURE_CCM_ENABLE
TYDefs.h, 128	Tylsp.h, 137
TY_INT_SGPM_EPI_CH1	TY_ISP_FEATURE_DEBUG_LOG
TYDefs.h, 128	Tylsp.h, 137
TY_INT_SGPM_EPI_HF	TY_ISP_FEATURE_ID
TYDefs.h, 128	Tylsp.h, 137
TY INT SGPM EPI THRESH	TY_ISP_FEATURE_IMAGE_SIZE
TYDefs.h, 128	Tylsp.h, 137
TY_INT_SGPM_NORMAL_PHASE_OFFSET	TY_ISP_FEATURE_INFO, 26
TYDefs.h, 128	TY_LASER_PARAM, 26
TY_INT_SGPM_NORMAL_PHASE_SCALE	TY_LASER_PATTERN_PARAM, 27
TYDefs.h, 128	TY_PHC_GROUP_ATTR, 28
TY_INT_SGPM_ORDER_FILTER_CHN	TY_PHC_GROUP_ATTR::phc_group_attr, 9
TYDefs.h, 128	TY_PIXEL_BITS_LIST
TY_INT_SGPM_PHASE_NUM	TYDefs.h, 124, 129
TYDefs.h, 128	TY_PIXEL_COLOR_DESC, 28
TY_INT_SGPM_REF_PHASE_OFFSET	TY_PIXEL_DESC, 29
TYDefs.h, 128	TY_PIXEL_FORMAT_BAYER8BG
TY_INT_SGPM_REF_PHASE_SCALE	TYDefs.h, 130
TYDefs.h, 128	TY_PIXEL_FORMAT_BAYER8GB
TY_INT_TOF_ANTI_SUNLIGHT_INDEX	TYDefs.h, 130
TYDefs.h, 129	TY_PIXEL_FORMAT_BAYER8GR
TY_INT_TOF_CHANNEL	TYDefs.h, 130
TYDefs.h, 129	TY_PIXEL_FORMAT_BAYER8RG
TY_INT_TOF_HDR_RATIO	TYDefs.h, 130
TYDefs.h, 127	TY_PIXEL_FORMAT_BGR
TY_INT_TOF_JITTER_THRESHOLD	TYDefs.h, 130
TYDefs.h, 127	TY_PIXEL_FORMAT_BGR48
TY_INT_TOF_MODULATION_THRESHOLD	TYDefs.h, 130
TYDefs.h, 129	TY_PIXEL_FORMAT_CSI_BAYER10BGGR
TY_INT_TRIGGER_DELAY_US	TYDefs.h, 130
TYDefs.h, 127	TY PIXEL FORMAT CSI BAYER10GBRG
TY_INT_TRIGGER_DURATION_US	TYDefs.h, 130
TYDefs.h, 127	TY_PIXEL_FORMAT_CSI_BAYER10GRBG
TY_INT_WIDTH	TYDefs.h, 130
TYDefs.h, 126	TY_PIXEL_FORMAT_CSI_BAYER10RGGB
TY_INTERFACE_INFO, 25	TYDefs.h, 130
TYDefs.h, 124	TY_PIXEL_FORMAT_CSI_BAYER12BGGR
TY_INTERFACE_TYPE_LIST	TYDefs.h, 130
TYDefs.h, 124, 129	TY_PIXEL_FORMAT_CSI_BAYER12GBRG
TY_ISP_FEATURE_AUTO_EXPOSURE_RANGE	TYDefs.h, 130
Tylsp.h, 137	TY_PIXEL_FORMAT_CSI_BAYER12GRBG
TY_ISP_FEATURE_AUTO_EXPOSURE_UPDATE_INTE	
Tylsp.h, 137	TY_PIXEL_FORMAT_CSI_BAYER12RGGB
· ·	
TY_ISP_FEATURE_AUTO_GAIN_RANGE	TYDefs.h, 130
Tylsp.h, 137	TY_PIXEL_FORMAT_CSI_MONO10
TY_ISP_FEATURE_BLACK_LEVEL	TYDefs.h, 130
Tylsp.h, 137	TY_PIXEL_FORMAT_CSI_MONO12

TYDefs.h, 130	TYDefs.h, 130
TY_PIXEL_FORMAT_DEPTH16	TY_RESOLUTION_MODE_640x360
TYDefs.h, 130	TYDefs.h, 130
TY_PIXEL_FORMAT_JPEG	TY_RESOLUTION_MODE_640x400
TYDefs.h, 130	TYDefs.h, 131
TY_PIXEL_FORMAT_LIST	TY_RESOLUTION_MODE_640x480
TYDefs.h, 129	TYDefs.h, 131
TY_PIXEL_FORMAT_MJPG	TY_RESOLUTION_MODE_800x600
TYDefs.h, 130	TYDefs.h, 131
TY_PIXEL_FORMAT_MONO	TY_RESOLUTION_MODE_960x1280
TYDefs.h, 130	TYDefs.h, 131
TY_PIXEL_FORMAT_MONO16	TY_RESOLUTION_MODE_LIST
TYDefs.h, 130	TYDefs.h, 130
TY PIXEL FORMAT RGB	TY STRUCT AEC ROI
TYDefs.h, 130	TYDefs.h, 127
TY PIXEL FORMAT RGB48	TY STRUCT CAM CALIB DATA
TYDefs.h, 130	TYDefs.h, 126
TY_PIXEL_FORMAT_TOF_IR_MONO16	TY_STRUCT_CAM_DISTORTION
TYDefs.h, 130	TYDefs.h, 126
TY_PIXEL_FORMAT_XYZ48	TY_STRUCT_CAM_INTRINSIC
TYDefs.h, 130	TYDefs.h, 126
TY_PIXEL_FORMAT_YUYV	TY_STRUCT_CAM_RECTIFIED_INTRI
TYDefs.h, 130	TYDefs.h, 126
TY_PIXEL_FORMAT_YVYU	TY_STRUCT_CAM_RECTIFIED_ROTATION
TYDefs.h, 130	TYDefs.h, 126
TY_RESOLUTION_MODE_1280x720	TY_STRUCT_CAM_STATISTICS
TYDefs.h, 131	TYDefs.h, 126
TY_RESOLUTION_MODE_1280x800	TY_STRUCT_DI0_WORKMODE
TYDefs.h, 131	TYDefs.h, 127
TY_RESOLUTION_MODE_1280x960	TY_STRUCT_DI1_WORKMODE
TYDefs.h, 131	TYDefs.h, 127
TY_RESOLUTION_MODE_1600x1200	TY STRUCT DI2 WORKMODE
TYDefs.h, 131	
	TYDets.h. 127
	TYDefs.h, 127 TY STRUCT DO0 WORKMODE
TY_RESOLUTION_MODE_160x100	TY_STRUCT_DO0_WORKMODE
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130	TY_STRUCT_DO0_WORKMODE TYDefs.h, 127
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130 TY_RESOLUTION_MODE_160x120	TY_STRUCT_DO0_WORKMODE TYDefs.h, 127 TY_STRUCT_DO1_WORKMODE
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130 TY_RESOLUTION_MODE_160x120 TYDefs.h, 130	TY_STRUCT_DO0_WORKMODE TYDefs.h, 127 TY_STRUCT_DO1_WORKMODE TYDefs.h, 127
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536	TY_STRUCT_DO0_WORKMODE TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536 TYDefs.h, 131	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536 TYDefs.h, 131  TY_RESOLUTION_MODE_240x320	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX
TY_RESOLUTION_MODE_160x100    TYDefs.h, 130  TY_RESOLUTION_MODE_160x120    TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080    TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440    TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536    TYDefs.h, 131  TY_RESOLUTION_MODE_240x320    TYDefs.h, 130	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536 TYDefs.h, 131  TY_RESOLUTION_MODE_240x320 TYDefs.h, 130  TY_RESOLUTION_MODE_240x96	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536 TYDefs.h, 131  TY_RESOLUTION_MODE_240x320 TYDefs.h, 130  TY_RESOLUTION_MODE_240x96 TYDefs.h, 131	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536 TYDefs.h, 131  TY_RESOLUTION_MODE_240x320 TYDefs.h, 130  TY_RESOLUTION_MODE_240x96 TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536 TYDefs.h, 131  TY_RESOLUTION_MODE_240x320 TYDefs.h, 130  TY_RESOLUTION_MODE_240x96 TYDefs.h, 131	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536 TYDefs.h, 131  TY_RESOLUTION_MODE_240x320 TYDefs.h, 130  TY_RESOLUTION_MODE_240x96 TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS
TY_RESOLUTION_MODE_160x100 TYDefs.h, 130  TY_RESOLUTION_MODE_160x120 TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080 TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440 TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536 TYDefs.h, 131  TY_RESOLUTION_MODE_240x320 TYDefs.h, 130  TY_RESOLUTION_MODE_240x96 TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920 TYDefs.h, 131	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS     TYDefs.h, 128
TY_RESOLUTION_MODE_160x100     TYDefs.h, 130  TY_RESOLUTION_MODE_160x120     TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080     TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440     TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536     TYDefs.h, 131  TY_RESOLUTION_MODE_240x320     TYDefs.h, 130  TY_RESOLUTION_MODE_240x96     TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920     TYDefs.h, 131  TY_RESOLUTION_MODE_2592x1944	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS     TYDefs.h, 128  TY_STRUCT_IMU_ACC_MISALIGNMENT
TY_RESOLUTION_MODE_160x100     TYDefs.h, 130  TY_RESOLUTION_MODE_160x120     TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080     TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440     TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536     TYDefs.h, 131  TY_RESOLUTION_MODE_240x320     TYDefs.h, 130  TY_RESOLUTION_MODE_240x96     TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920     TYDefs.h, 131  TY_RESOLUTION_MODE_2592x1944     TYDefs.h, 131	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS     TYDefs.h, 128  TY_STRUCT_IMU_ACC_MISALIGNMENT     TYDefs.h, 128
TY_RESOLUTION_MODE_160x100     TYDefs.h, 130  TY_RESOLUTION_MODE_160x120     TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080     TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440     TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536     TYDefs.h, 131  TY_RESOLUTION_MODE_240x320     TYDefs.h, 130  TY_RESOLUTION_MODE_240x96     TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920     TYDefs.h, 131  TY_RESOLUTION_MODE_2592x1944     TYDefs.h, 131  TY_RESOLUTION_MODE_320x180     TYDefs.h, 130	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS     TYDefs.h, 128  TY_STRUCT_IMU_ACC_MISALIGNMENT     TYDefs.h, 128  TY_STRUCT_IMU_ACC_SCALE     TYDefs.h, 128
TY_RESOLUTION_MODE_160x100     TYDefs.h, 130  TY_RESOLUTION_MODE_160x120     TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080     TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440     TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536     TYDefs.h, 131  TY_RESOLUTION_MODE_240x320     TYDefs.h, 130  TY_RESOLUTION_MODE_240x96     TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920     TYDefs.h, 131  TY_RESOLUTION_MODE_2592x1944     TYDefs.h, 131  TY_RESOLUTION_MODE_320x180     TYDefs.h, 130  TY_RESOLUTION_MODE_320x200	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS     TYDefs.h, 128  TY_STRUCT_IMU_ACC_MISALIGNMENT     TYDefs.h, 128  TY_STRUCT_IMU_ACC_SCALE     TYDefs.h, 128  TY_STRUCT_IMU_CAM_TO_IMU
TY_RESOLUTION_MODE_160x100     TYDefs.h, 130  TY_RESOLUTION_MODE_160x120     TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080     TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440     TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536     TYDefs.h, 131  TY_RESOLUTION_MODE_240x320     TYDefs.h, 130  TY_RESOLUTION_MODE_240x96     TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920     TYDefs.h, 131  TY_RESOLUTION_MODE_2592x1944     TYDefs.h, 131  TY_RESOLUTION_MODE_320x180     TYDefs.h, 130  TY_RESOLUTION_MODE_320x200     TYDefs.h, 130	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS     TYDefs.h, 128  TY_STRUCT_IMU_ACC_MISALIGNMENT     TYDefs.h, 128  TY_STRUCT_IMU_ACC_SCALE     TYDefs.h, 128  TY_STRUCT_IMU_ACC_SCALE     TYDefs.h, 128  TY_STRUCT_IMU_CAM_TO_IMU     TYDefs.h, 128
TY_RESOLUTION_MODE_160x100     TYDefs.h, 130  TY_RESOLUTION_MODE_160x120     TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080     TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440     TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536     TYDefs.h, 131  TY_RESOLUTION_MODE_240x320     TYDefs.h, 130  TY_RESOLUTION_MODE_240x96     TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920     TYDefs.h, 131  TY_RESOLUTION_MODE_2592x1944     TYDefs.h, 131  TY_RESOLUTION_MODE_320x180     TYDefs.h, 130  TY_RESOLUTION_MODE_320x200     TYDefs.h, 130  TY_RESOLUTION_MODE_320x240	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS     TYDefs.h, 128  TY_STRUCT_IMU_ACC_MISALIGNMENT     TYDefs.h, 128  TY_STRUCT_IMU_ACC_SCALE     TYDefs.h, 128  TY_STRUCT_IMU_CAM_TO_IMU     TYDefs.h, 128  TY_STRUCT_IMU_CAM_TO_IMU     TYDefs.h, 128  TY_STRUCT_IMU_GYRO_BIAS
TY_RESOLUTION_MODE_160x100     TYDefs.h, 130  TY_RESOLUTION_MODE_160x120     TYDefs.h, 130  TY_RESOLUTION_MODE_1920x1080     TYDefs.h, 131  TY_RESOLUTION_MODE_1920x1440     TYDefs.h, 131  TY_RESOLUTION_MODE_2048x1536     TYDefs.h, 131  TY_RESOLUTION_MODE_240x320     TYDefs.h, 130  TY_RESOLUTION_MODE_240x96     TYDefs.h, 131  TY_RESOLUTION_MODE_2560x1920     TYDefs.h, 131  TY_RESOLUTION_MODE_2592x1944     TYDefs.h, 131  TY_RESOLUTION_MODE_320x180     TYDefs.h, 130  TY_RESOLUTION_MODE_320x200     TYDefs.h, 130	TY_STRUCT_DO0_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO1_WORKMODE     TYDefs.h, 127  TY_STRUCT_DO2_WORKMODE     TYDefs.h, 127  TY_STRUCT_EXTRINSIC_TO_DEPTH     TYDefs.h, 126  TY_STRUCT_EXTRINSIC_TO_IR_LEFT     TYDefs.h, 126  TY_STRUCT_FLOOD_ENABLE_BY_IDX     TYDefs.h, 127  TY_STRUCT_FLOOD_POWER_BY_IDX     TYDefs.h, 127  TY_STRUCT_IMU_ACC_BIAS     TYDefs.h, 128  TY_STRUCT_IMU_ACC_MISALIGNMENT     TYDefs.h, 128  TY_STRUCT_IMU_ACC_SCALE     TYDefs.h, 128  TY_STRUCT_IMU_ACC_SCALE     TYDefs.h, 128  TY_STRUCT_IMU_CAM_TO_IMU     TYDefs.h, 128

TYDefs.h, 128	TYGetByteArrayAttr, 49
TY_STRUCT_IMU_GYRO_SCALE	TYGetByteArraySize, 51
TYDefs.h, 128	TYGetComponentIDs, 52
TY_STRUCT_LASER_ENABLE_BY_IDX	TYGetDeviceFeatureInfo, 53
TYDefs.h, 127	TYGetDeviceFeatureNumber, 54
TY_STRUCT_LASER_POWER_BY_IDX	TYGetDeviceInfo, 55
TYDefs.h, 127	TYGetDeviceInterface, 56
TY_STRUCT_PHC_GROUP_ATTR	TYGetDeviceList, 57
TYDefs.h, 128	TYGetDeviceNumber, 57
TY_STRUCT_TOF_FREQ	TYGetDeviceXML, 58 TYGetDeviceXMLSize, 59
TYDefs.h, 129 TY_STRUCT_TRIGGER_PARAM	•
TYDefs.h, 127	TYGetEnabledComponents, 60 TYGetEnum, 61
TY_STRUCT_TRIGGER_PARAM_EX	TYGetEnumEntryCount, 62
TYDefs.h, 127	TYGetEnumEntryInfo, 63
TY_STRUCT_TRIGGER_TIMER_LIST	TYGetFeatureInfo, 64
TYDefs.h, 127	TYGetFloat, 65
TY_STRUCT_TRIGGER_TIMER_PERIOD	TYGetFloatRange, 67
TYDefs.h, 127	TYGetFrameBufferSize, 68
TY TEMP DATA, 29	TYGetInt, 69
TY TOF FREQ, 30	TYGetInterfaceList, 70
TY_TRIGGER_MODE_LIST	TYGetInterfaceNumber, 71
TYDefs.h, 125, 131	TYGetIntRange, 72
TY_TRIGGER_MODE_M_PER	TYGetString, 73
TYDefs.h, 131	TYGetStringLength, 74
TY_TRIGGER_MODE_M_SIG	TYGetStruct, 76
TYDefs.h, 131	TYHasDevice, 78
TY_TRIGGER_MODE_OFF	TYHasFeature, 79
TYDefs.h, 131	TYHasInterface, 80
TY_TRIGGER_MODE_PER_PASS	TYLibVersion, 80
TYDefs.h, 131	TYOpenDevice, 81
TY_TRIGGER_MODE_PER_PASS2	TYOpenDeviceWithIP, 82
TYDefs.h, 131	TYOpenInterface, 84
TY_TRIGGER_MODE_SIG_PASS	TYRegisterEventCallback, 84
TYDefs.h, 131	TYRegisterImuCallback, 85
TY_TRIGGER_MODE_SLAVE	TYRemoveLogFile, 86
TYDefs.h, 131	TYRemoveLogServer, 86
TY_TRIGGER_PARAM, 30	TYSetBool 88
TY_TRIGGER_PARAM_EX, 30	TYSetBute Arroy 80
TY_TRIGGER_TIMER_LIST, 31 TY_TRIGGER_TIMER_PERIOD, 31	TYSetByteArray, 89 TYSetEnum, 91
TY VECT 3F, 31	TYSetFloat, 92
TY_VERSION_INFO, 32	TYSetIndt, 94
TYApi.h, 33	TYSetLogLevel, 95
TYAppendLogToFile, 37	TYSetLogPrefix, 95
TYAppendLogToServer, 38	TYSetString, 96
TYClearBufferQueue, 39	TYSetStruct, 97
TYCloseDevice, 39	TYStartCapture, 99
TYCloseInterface, 40	TYStopCapture, 100
TYDeinitLib, 40	TYUpdateAllDeviceList, 101
TYDisableComponents, 41	TYUpdateDeviceList, 101
TYEnableComponents, 42	TYUpdateInterfaceList, 102
TYEnqueueBuffer, 43	TYAppendLogToFile
TYErrorString, 44	TYApi.h, 37
TYFetchFrame, 44	TYAppendLogToServer
TYForceDeviceIP, 45	TYApi.h, 38
TYGetBool, 46	TYClearBufferQueue
TYGetByteArray, 48	TYApi.h, 39

TYCloseDevice	TY_COMPONENT_RGB_CAM_LEFT, 126
TYApi.h, 39	TY_COMPONENT_RGB_CAM_RIGHT, 126
TYCloseInterface	TY_COMPONENT_STORAGE, 126
TYApi.h, 40	TY_DECLARE_IMAGE_MODE1, 118
TYCoordinateMapper.h, 102	TY_DEVICE_BASE_INFO, 122
TYDepthImageFillEmptyRegion, 105	TY_DEVICE_COMPONENT_LIST, 122, 125
TYInvertExtrinsic, 105	TY_ENUM_DEPTH_QUALITY, 129
TYMAP_CHECKRET, 104	TY_ENUM_ENTRY, 122
TYMapDepthImageToPoint3d, 106	TY_ENUM_IMAGE_MODE, 126
TYMapDepthToPoint3d, 106	TY_ENUM_IMU_FPS, 128
TYMapPoint3dToDepth, 107	TY_ENUM_STREAM_ASYNC, 127
TYMapPoint3dToDepthImage, 107	TY_ENUM_TIME_SYNC_TYPE, 127
TYMapPoint3dToPoint3d, 108	TY_ENUM_TRIGGER_POL, 126
•	
TYDefs.h, 108	TY_FEATURE_ID, 123
TY_ACC_BIAS, 119	TY_FEATURE_ID_LIST, 126
TY_ACC_MISALIGNMENT, 119	TY_FLOAT_EXPOSURE_TIME_US, 127
TY_ACC_SCALE, 119	TY_FLOAT_RANGE, 123
TY_ACCESS_MODE_LIST, 119, 125	TY_FLOAT_SCALE_UNIT, 126
TY_BOOL_AUTO_AWB, 127	TY_FLOAT_SGPM_EPI_HS, 128
TY_BOOL_AUTO_EXPOSURE, 127	TY_GYRO_BIAS, 123
TY_BOOL_AUTO_GAIN, 127	TY_GYRO_MISALIGNMENT, 124
TY_BOOL_BRIGHTNESS_HISTOGRAM, 127	TY_GYRO_SCALE, 124
TY_BOOL_CMOS_SYNC, 127	TY_INT_ANALOG_GAIN, 127
TY_BOOL_DEPTH_POSTPROC, 127	TY_INT_B_GAIN, 127
TY_BOOL_HDR, 128	TY_INT_CAPTURE_TIME_US, 127
TY_BOOL_IMU_DATA_ONOFF, 128	TY_INT_DEPTH_MAX_MM, 128
TY_BOOL_IR_FLASHLIGHT, 127	TY_INT_DEPTH_MIN_MM, 128
TY_BOOL_KEEP_ALIVE_ONOFF, 127	TY_INT_EXPOSURE_TIME, 127
TY_BOOL_LASER_AUTO_CTRL, 127	TY_INT_FILTER_THRESHOLD, 129
TY_BOOL_RGB_FLASHLIGHT, 127	TY_INT_FRAME_PER_TRIGGER, 126
TY_BOOL_SGBM_HFILTER_HALF_WIN, 128	TY_INT_G_GAIN, 127
TY_BOOL_SGBM_LRC, 128	TY_INT_GAIN, 127
TY_BOOL_SGBM_MEDFILTER, 128	TY_INT_HEIGHT, 126
TY_BOOL_SGPM_EPI_EN, 128	TY INT IR FLASHLIGHT INTENSITY, 127
TY_BOOL_SGPM_ORDER_FILTER_EN, 128	TY_INT_KEEP_ALIVE_TIMEOUT, 127
TY_BOOL_TIME_SYNC_READY, 127	TY_INT_LASER_POWER, 127
TY_BOOL_TOF_ANTI_INTERFERENCE, 129	TY INT LINK CMD TIMEOUT, 126
TY_BOOL_TRIGGER_OUT_IO, 127	TY INT MAX SPECKLE DIFF, 129
TY_BOOL_UNDISTORTION, 127	TY_INT_MAX_SI ECKLE_DII1, 129 TY_INT_MAX_SPECKLE_SIZE, 129
	TY_INT_NTP_SERVER_IP, 126
TY_BYTEARRAY_ATTR, 120	
TY_BYTEARRAY_CUSTOM_BLOCK, 126	TY_INT_PACKET_DELAY, 126
TY_BYTEARRAY_HDR_PARAMETER, 128	TY_INT_R_GAIN, 127
TY_BYTEARRAY_ISP_BLOCK, 126	TY_INT_RGB_FLASHLIGHT_INTENSITY, 127
TY_CAMERA_CALIB_INFO, 120	TY_INT_SGBM_DISPARITY_NUM, 128
TY_CAMERA_DISTORTION, 120	TY_INT_SGBM_DISPARITY_OFFSET, 128
TY_CAMERA_EXTRINSIC, 120	TY_INT_SGBM_IMAGE_NUM, 128
TY_CAMERA_INTRINSIC, 121	TY_INT_SGBM_LRC_DIFF, 128
TY_CAMERA_ROTATION, 121	TY_INT_SGBM_MATCH_WIN_HEIGHT, 128
TY_CAMERA_TO_IMU, 121	TY_INT_SGBM_MATCH_WIN_WIDTH, 128
TY_COMPONENT_BRIGHT_HISTO, 126	TY_INT_SGBM_MEDFILTER_THRESH, 128
TY_COMPONENT_DEPTH_CAM, 126	TY_INT_SGBM_SEMI_PARAM_P1, 128
TY_COMPONENT_DEVICE, 126	TY_INT_SGBM_SEMI_PARAM_P1_SCALE, 128
TY_COMPONENT_ID, 122	TY_INT_SGBM_SEMI_PARAM_P2, 128
TY_COMPONENT_IMU, 126	TY_INT_SGBM_TEXTURE_OFFSET, 128
TY_COMPONENT_IR_CAM_LEFT, 126	TY_INT_SGBM_TEXTURE_THRESH, 128
TY_COMPONENT_IR_CAM_RIGHT, 126	TY_INT_SGBM_UNIQUE_ABSDIFF, 128
TY_COMPONENT_LASER, 126	TY_INT_SGBM_UNIQUE_FACTOR, 128
TY_COMPONENT_RGB_CAM, 126	TY_INT_SGBM_UNIQUE_MAX_COST, 128

TY_INT_SGPM_EPI_CH0, 128	TY_RESOLUTION_MODE_240x320, 130
TY_INT_SGPM_EPI_CH1, 128	TY_RESOLUTION_MODE_240x96, 131
TY_INT_SGPM_EPI_HF, 128	TY_RESOLUTION_MODE_2560x1920, 131
TY INT SGPM EPI THRESH, 128	TY RESOLUTION MODE 2592x1944, 131
TY INT SGPM NORMAL PHASE OFFSET, 128	TY_RESOLUTION_MODE_320x180, 130
TY_INT_SGPM_NORMAL_PHASE_SCALE, 128	TY_RESOLUTION_MODE_320x200, 130
TY_INT_SGPM_ORDER_FILTER_CHN, 128	TY_RESOLUTION_MODE_320x240, 130
TY_INT_SGPM_PHASE_NUM, 128	TY_RESOLUTION_MODE_480x640, 130
TY_INT_SGPM_REF_PHASE_OFFSET, 128	TY_RESOLUTION_MODE_640x360, 130
TY_INT_SGPM_REF_PHASE_SCALE, 128	TY_RESOLUTION_MODE_640x400, 131
TY_INT_TOF_ANTI_SUNLIGHT_INDEX, 129	TY_RESOLUTION_MODE_640x480, 131
TY_INT_TOF_CHANNEL, 129	TY_RESOLUTION_MODE_800x600, 131
TY_INT_TOF_HDR_RATIO, 127	TY_RESOLUTION_MODE_960x1280, 131
TY_INT_TOF_JITTER_THRESHOLD, 127	TY_RESOLUTION_MODE_LIST, 130
TY_INT_TOF_MODULATION_THRESHOLD, 129	TY_STRUCT_AEC_ROI, 127
TY_INT_TRIGGER_DELAY_US, 127	TY_STRUCT_CAM_CALIB_DATA, 126
TY_INT_TRIGGER_DURATION_US, 127	TY_STRUCT_CAM_DISTORTION, 126
TY_INT_WIDTH, 126	TY_STRUCT_CAM_INTRINSIC, 126
TY_INTERFACE_INFO, 124	TY_STRUCT_CAM_RECTIFIED_INTRI, 126
TY_INTERFACE_TYPE_LIST, 124, 129	TY_STRUCT_CAM_RECTIFIED_ROTATION, 126
TY_PIXEL_BITS_LIST, 124, 129	TY_STRUCT_CAM_STATISTICS, 126
TY_PIXEL_FORMAT_BAYER8BG, 130	TY_STRUCT_DI0_WORKMODE, 127
TY_PIXEL_FORMAT_BAYER8GB, 130	TY_STRUCT_DI1_WORKMODE, 127
TY PIXEL FORMAT BAYER8GR, 130	TY STRUCT DI2 WORKMODE, 127
TY_PIXEL_FORMAT_BAYER8RG, 130	TY_STRUCT_DO0_WORKMODE, 127
TY_PIXEL_FORMAT_BGR, 130	TY_STRUCT_DO1_WORKMODE, 127
TY_PIXEL_FORMAT_BGR48, 130	TY_STRUCT_DO2_WORKMODE, 127
TY_PIXEL_FORMAT_CSI_BAYER10BGGR, 130	TY_STRUCT_EXTRINSIC_TO_DEPTH, 126
TY_PIXEL_FORMAT_CSI_BAYER10GBRG, 130	TY_STRUCT_EXTRINSIC_TO_IR_LEFT, 126
TY_PIXEL_FORMAT_CSI_BAYER10GRBG, 130	TY_STRUCT_FLOOD_ENABLE_BY_IDX, 127
TY_PIXEL_FORMAT_CSI_BAYER10RGGB, 130	TY_STRUCT_FLOOD_POWER_BY_IDX, 127
TY_PIXEL_FORMAT_CSI_BAYER12BGGR, 130	TY_STRUCT_IMU_ACC_BIAS, 128
TY_PIXEL_FORMAT_CSI_BAYER12GBRG, 130	TY_STRUCT_IMU_ACC_MISALIGNMENT, 128
TY PIXEL FORMAT CSI BAYER12GRBG, 130	TY STRUCT IMU ACC SCALE, 128
TY_PIXEL_FORMAT_CSI_BAYER12RGGB, 130	TY_STRUCT_IMU_CAM_TO_IMU, 128
TY_PIXEL_FORMAT_CSI_MONO10, 130	TY_STRUCT_IMU_GYRO_BIAS, 128
TY_PIXEL_FORMAT_CSI_MONO12, 130	TY STRUCT IMU GYRO MISALIGNMENT, 128
TY_PIXEL_FORMAT_DEPTH16, 130	TY_STRUCT_IMU_GYRO_SCALE, 128
TY_PIXEL_FORMAT_JPEG, 130	TY_STRUCT_LASER_ENABLE_BY_IDX, 127
TY_PIXEL_FORMAT_LIST, 129	TY_STRUCT_LASER_POWER_BY_IDX, 127
TY_PIXEL_FORMAT_MJPG, 130	TY_STRUCT_PHC_GROUP_ATTR, 128
TY_PIXEL_FORMAT_MONO, 130	TY_STRUCT_TOF_FREQ, 129
TY_PIXEL_FORMAT_MONO16, 130	TY_STRUCT_TRIGGER_PARAM, 127
TY_PIXEL_FORMAT_RGB, 130	TY_STRUCT_TRIGGER_PARAM_EX, 127
TY_PIXEL_FORMAT_RGB48, 130	TY_STRUCT_TRIGGER_TIMER_LIST, 127
TY_PIXEL_FORMAT_TOF_IR_MONO16, 130	TY_STRUCT_TRIGGER_TIMER_PERIOD, 127
TY_PIXEL_FORMAT_XYZ48, 130	TY_TRIGGER_MODE_LIST, 125, 131
TY_PIXEL_FORMAT_YUYV, 130	TY_TRIGGER_MODE_M_PER, 131
TY_PIXEL_FORMAT_YVYU, 130	TY_TRIGGER_MODE_M_SIG, 131
TY_RESOLUTION_MODE_1280x720, 131	TY_TRIGGER_MODE_OFF, 131
TY_RESOLUTION_MODE_1280x800, 131	TY_TRIGGER_MODE_PER_PASS, 131
TY_RESOLUTION_MODE_1280x960, 131	TY_TRIGGER_MODE_PER_PASS2, 131
TY_RESOLUTION_MODE_1600x1200, 131	TY_TRIGGER_MODE_SIG_PASS, 131
TY_RESOLUTION_MODE_160x100, 130	TY_TRIGGER_MODE_SLAVE, 131
TY_RESOLUTION_MODE_160x120, 130	TYDeinitLib
TY_RESOLUTION_MODE_1920x1080, 131	TYApi.h, 40
TY_RESOLUTION_MODE_1920x1440, 131	TYDepthEnhenceFilter
TY_RESOLUTION_MODE_2048x1536, 131	TYImageProc.h, 133
1 _ 1 _ 1 _ 0 0 _ 0 1 0 1 _ W 0 D L _ 2 0 T 0 X 1 0 0 0 , 10 1	i i illiagor room, roo

TYDepthImageFillEmptyRegion	TYGetInt
TYCoordinateMapper.h, 105	TYApi.h, 69 TYGetInterfaceList
TYDepthSpeckleFilter	
TYImageProc.h, 133	TYApi.h, 70
TYDisableComponents	TYGetInterfaceNumber
TYApi.h, 41	TYApi.h, 71
TYEnableComponents	TYGetIntRange
TYApi.h, 42	TYApi.h, 72
TYEnqueueBuffer	TYGetString
TYApi.h, 43	TYApi.h, 73
TYErrorString	TYGetStringLength
TYApi.h, 44	TYApi.h, 74
TYFetchFrame	TYGetStruct
TYApi.h, 44	TYApi.h, 76
TYForceDeviceIP	TYHasDevice
TYApi.h, 45	TYApi.h, 78
TYGetBool	TYHasFeature
TYApi.h, 46	TYApi.h, 79
TYGetByteArray	TYHasInterface
TYApi.h, 48	TYApi.h, 80
TYGetByteArrayAttr	TYImageProc.h, 131
TYApi.h, 49	TYDepthEnhenceFilter, 133
TYGetByteArraySize	TYDepthSpeckleFilter, 133
TYApi.h, 51	TYImageProcesAcceEnable, 134
TYGetComponentIDs	TYUndistortImage, 134
TYApi.h, 52	TYImageProcesAcceEnable
TYGetDeviceFeatureInfo	TYImageProc.h, 134
TYApi.h, 53	TYInvertExtrinsic
TYGetDeviceFeatureNumber	TYCoordinateMapper.h, 105
TYApi.h, 54	Tylsp.h, 135
TYGetDeviceInfo	TY_ISP_FEATURE_AUTO_EXPOSURE_RANGE,
TYApi.h, 55	137
TYGetDeviceInterface	TY ISP FEATURE AUTO EXPOSURE UPDATE INTERVAL,
TYApi.h, 56	137
TYGetDeviceList	TY_ISP_FEATURE_AUTO_GAIN_RANGE, 137
TYApi.h, 57	TY_ISP_FEATURE_BLACK_LEVEL, 137
TYGetDeviceNumber	TY_ISP_FEATURE_BLACK_LEVEL_COLUMN,
TYApi.h, 57	137
TYGetDeviceXML	TY_ISP_FEATURE_BLACK_LEVEL_GAIN, 137
TYApi.h, 58	TY ISP FEATURE BLACK LEVEL GAIN COLUMN,
TYGetDeviceXMLSize	137
TYApi.h, 59	TY_ISP_FEATURE_CAM_DEV_COMPONENT,
TYGetEnabledComponents	137
TYApi.h, 60	TY_ISP_FEATURE_CAM_DEV_HANDLE, 137
TYGetEnum	TY ISP FEATURE CCM ENABLE, 137
TYApi.h, 61	TY_ISP_FEATURE_DEBUG_LOG, 137
•	TY_ISP_FEATURE_ID, 137
TYAnih 60	TY_ISP_FEATURE_ID, 137 TY_ISP_FEATURE_IMAGE_SIZE, 137
TYApi.h, 62	
TYGetEnumEntryInfo	TYLibVersion
TYApi.h, 63	TYAPP OUEOGRET
TYGetFeatureInfo	TYMAP_CHECKRET
TYApi.h, 64	TYCoordinateMapper.h, 104
TYGetFloat	TYMapDepthImageToPoint3d
TYApi.h, 65	TYCoordinateMapper.h, 106
TYGetFloatRange	TYMapDepthToPoint3d
TYApi.h, 67	TYCoordinateMapper.h, 106
TYGetFrameBufferSize	TYMapPoint3dToDepth
TYApi.h, 68	TYCoordinateMapper.h, 107

```
TYMapPoint3dToDepthImage
    TYCoordinateMapper.h, 107
TYMapPoint3dToPoint3d
    TYCoordinateMapper.h, 108
TYOpenDevice
    TYApi.h, 81
TYOpenDeviceWithIP
    TYApi.h, 82
TYOpenInterface
    TYApi.h, 84
TYRegisterEventCallback
    TYApi.h, 84
TYRegisterImuCallback
    TYApi.h, 85
TYRemoveLogFile
    TYApi.h, 86
TYRemoveLogServer
    TYApi.h, 86
TYSendSoftTrigger
    TYApi.h, 87
TYSetBool
    TYApi.h, 88
TYSetByteArray
    TYApi.h, 89
TYSetEnum
    TYApi.h, 91
TYSetFloat
    TYApi.h, 92
TYSetInt
    TYApi.h, 94
TYSetLogLevel
    TYApi.h, 95
TYSetLogPrefix
    TYApi.h, 95
TYSetString
    TYApi.h, 96
TYSetStruct
    TYApi.h, 97
TYStartCapture
    TYApi.h, 99
TYStopCapture
    TYApi.h, 100
TYUndistortImage
    TYImageProc.h, 134
TYUpdateAllDeviceList
    TYApi.h, 101
TYUpdateDeviceList
    TYApi.h, 101
TYUpdateInterfaceList
    TYApi.h, 102
unit size
    TY_BYTEARRAY_ATTR, 11
valid size
    TY_BYTEARRAY_ATTR, 12
```