

TYCampport3

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 DepthEnhanceParameters Struct Reference	7
4.1.1 Detailed Description	7
4.2 DepthSpeckleFilterParameters Struct Reference	7
4.2.1 Detailed Description	7
4.3 gz_header_s Struct Reference	8
4.3.1 Detailed Description	8
4.4 gzFile_s Struct Reference	8
4.4.1 Detailed Description	8
4.5 pattern_bin_param Struct Reference	8
4.5.1 Detailed Description	9
4.6 pattern_gray_param Struct Reference	9
4.6.1 Detailed Description	9
4.7 pattern_sine_param Struct Reference	9
4.7.1 Detailed Description	9
4.8 TY_PHC_GROUP_ATTR::phc_group_attr Struct Reference	10
4.8.1 Detailed Description	10
4.9 TY_ACC_BIAS Struct Reference	10
4.9.1 Detailed Description	10
4.10 TY_ACC_MISALIGNMENT Struct Reference	10
4.10.1 Detailed Description	11
4.11 TY_ACC_SCALE Struct Reference	11
4.11.1 Detailed Description	11
4.12 TY_AEC_ROI_PARAM Struct Reference	12
4.12.1 Detailed Description	12
4.13 TY_BYTEARRAY_ATTR Struct Reference	12
4.13.1 Detailed Description	12
4.13.2 Member Data Documentation	12
4.13.2.1 unit_size	13
4.13.2.2 valid_size	13
4.14 TY_CAMERA_CALIB_INFO Struct Reference	13
4.14.1 Detailed Description	14
4.15 TY_CAMERA_DISTORTION Struct Reference	14
4.15.1 Detailed Description	14

4.16 TY_CAMERA_EXTRINSIC Struct Reference	14
4.16.1 Detailed Description	14
4.17 TY_CAMERA_INTRINSIC Struct Reference	15
4.17.1 Detailed Description	15
4.18 TY_CAMERA_ROTATION Struct Reference	16
4.18.1 Detailed Description	16
4.19 TY_CAMERA_STATISTICS Struct Reference	16
4.19.1 Detailed Description	16
4.20 TY_CAMERA_TO_IMU Struct Reference	17
4.20.1 Detailed Description	17
4.21 TY_DEVICE_BASE_INFO Struct Reference	17
4.21.1 Detailed Description	18
4.22 TY_DEVICE_NET_INFO Struct Reference	18
4.22.1 Detailed Description	19
4.23 TY_DEVICE_USB_INFO Struct Reference	19
4.23.1 Detailed Description	19
4.24 TY_DI_WORKMODE Struct Reference	19
4.24.1 Detailed Description	19
4.25 TY_DO_WORKMODE Struct Reference	20
4.25.1 Detailed Description	20
4.26 TY_ENUM_ENTRY Struct Reference	20
4.26.1 Detailed Description	20
4.27 TY_EVENT_INFO Struct Reference	21
4.27.1 Detailed Description	21
4.28 TY_FEATURE_INFO Struct Reference	21
4.28.1 Detailed Description	21
4.29 TY_FLOAT_RANGE Struct Reference	22
4.29.1 Detailed Description	22
4.30 TY_FRAME_DATA Struct Reference	22
4.30.1 Detailed Description	23
4.31 TY_GYRO_BIAS Struct Reference	23
4.31.1 Detailed Description	23
4.32 TY_GYRO_MISALIGNMENT Struct Reference	23
4.32.1 Detailed Description	24
4.33 TY_GYRO_SCALE Struct Reference	24
4.33.1 Detailed Description	24
4.34 TY_IMAGE_DATA Struct Reference	25
4.34.1 Detailed Description	25
4.35 TY_IMU_DATA Struct Reference	25
4.35.1 Detailed Description	26
4.36 TY_INT_RANGE Struct Reference	26
4.36.1 Detailed Description	26

4.37 TY_INTERFACE_INFO Struct Reference	26
4.37.1 Detailed Description	27
4.38 TY_LASER_PARAM Struct Reference	27
4.38.1 Detailed Description	27
4.39 TY_LASER_PATTERN_PARAM Struct Reference	27
4.39.1 Detailed Description	28
4.40 TY_PHC_GROUP_ATTR Struct Reference	28
4.40.1 Detailed Description	29
4.41 TY_PIXEL_COLOR_DESC Struct Reference	29
4.41.1 Detailed Description	29
4.42 TY_PIXEL_DESC Struct Reference	29
4.42.1 Detailed Description	29
4.43 TY_TEMP_DATA Struct Reference	30
4.43.1 Detailed Description	30
4.44 TY_TOF_FREQ Struct Reference	30
4.44.1 Detailed Description	30
4.45 TY_TRIGGER_PARAM Struct Reference	30
4.45.1 Detailed Description	30
4.46 TY_TRIGGER_PARAM_EX Struct Reference	31
4.46.1 Detailed Description	31
4.47 TY_TRIGGER_TIMER_LIST Struct Reference	31
4.47.1 Detailed Description	31
4.48 TY_TRIGGER_TIMER_PERIOD Struct Reference	32
4.48.1 Detailed Description	32
4.49 TY_VECT_3F Struct Reference	32
4.49.1 Detailed Description	32
4.50 TY_VERSION_INFO Struct Reference	32
4.50.1 Detailed Description	33
4.51 TYEnumEntry Struct Reference	33
4.51.1 Detailed Description	33
4.52 z_stream_s Struct Reference	33
4.52.1 Detailed Description	33
5 File Documentation	35
5.1 TYApi.h File Reference	35
5.1.1 Detailed Description	39
5.1.2 Function Documentation	39
5.1.2.1 TYAppendLogToFile()	40
5.1.2.2 TYAppendLogToServer()	40
5.1.2.3 TYClearBufferQueue()	41
5.1.2.4 TYCloseDevice()	41
5.1.2.5 TYCloseInterface()	42

5.1.2.6 TYDeinitLib()	43
5.1.2.7 TYDisableComponents()	43
5.1.2.8 TYEnableComponents()	44
5.1.2.9 TYEnqueueBuffer()	45
5.1.2.10 TYErrorString()	46
5.1.2.11 TYFetchFrame()	46
5.1.2.12 TYForceDeviceIP()	47
5.1.2.13 TYGetBool()	49
5.1.2.14 TYGetByteArray()	50
5.1.2.15 TYGetByteArrayAttr()	51
5.1.2.16 TYGetByteArraySize()	53
5.1.2.17 TYGetComponentIDs()	54
5.1.2.18 TYGetDeviceFeatureInfo()	55
5.1.2.19 TYGetDeviceFeatureNumber()	56
5.1.2.20 TYGetDeviceInfo()	57
5.1.2.21 TYGetDeviceInterface()	58
5.1.2.22 TYGetDeviceList()	59
5.1.2.23 TYGetDeviceNumber()	60
5.1.2.24 TYGetDeviceXML()	60
5.1.2.25 TYGetDeviceXMLSize()	61
5.1.2.26 TYGetEnabledComponents()	62
5.1.2.27 TYGetEnum()	63
5.1.2.28 TYGetEnumEntryCount()	64
5.1.2.29 TYGetEnumEntryInfo()	65
5.1.2.30 TYGetFeatureInfo()	67
5.1.2.31 TYGetFloat()	68
5.1.2.32 TYGetFloatRange()	69
5.1.2.33 TYGetFrameBufferSize()	70
5.1.2.34 TYGetInt()	71
5.1.2.35 TYGetInterfaceList()	73
5.1.2.36 TYGetInterfaceNumber()	73
5.1.2.37 TYGetIntRange()	74
5.1.2.38 TYGetString()	75
5.1.2.39 TYGetStringLength()	76
5.1.2.40 TYGetStruct()	79
5.1.2.41 TYHasDevice()	80
5.1.2.42 TYHasFeature()	81
5.1.2.43 TYHasInterface()	82
5.1.2.44 TYLibVersion()	83
5.1.2.45 TYOpenDevice()	83
5.1.2.46 TYOpenDeviceWithIP()	85
5.1.2.47 TYOpenInterface()	86

5.1.2.48 TYRegisterEventCallback()	87
5.1.2.49 TYRegisterImuCallback()	87
5.1.2.50 TYRemoveLogFile()	88
5.1.2.51 TYRemoveLogServer()	88
5.1.2.52 TYSendSoftTrigger()	89
5.1.2.53 TYSetBool()	90
5.1.2.54 TYSetByteArray()	91
5.1.2.55 TYSetEnum()	93
5.1.2.56 TYSetFloat()	94
5.1.2.57 TYSetInt()	96
5.1.2.58 TYSetLogLevel()	97
5.1.2.59 TYSetLogPrefix()	97
5.1.2.60 TYSetString()	98
5.1.2.61 TYSetStruct()	99
5.1.2.62 TYStartCapture()	101
5.1.2.63 TYStopCapture()	102
5.1.2.64 TYUpdateAllDeviceList()	103
5.1.2.65 TYUpdateDeviceList()	103
5.1.2.66 TYUpdateInterfaceList()	104
5.2 TYCoordinateMapper.h File Reference	104
5.2.1 Detailed Description	106
5.2.2 Macro Definition Documentation	106
5.2.2.1 TYMAP_CHECKRET	107
5.2.3 Function Documentation	107
5.2.3.1 TYDepthImageFillEmptyRegion()	107
5.2.3.2 TYInvertExtrinsic()	107
5.2.3.3 TYMapDepthImageToPoint3d()	108
5.2.3.4 TYMapDepthToPoint3d()	108
5.2.3.5 TYMapPoint3dToDepth()	109
5.2.3.6 TYMapPoint3dToDepthImage()	109
5.2.3.7 TYMapPoint3dToPoint3d()	110
5.3 TYDefs.h File Reference	110
5.3.1 Detailed Description	120
5.3.2 Macro Definition Documentation	121
5.3.2.1 TY_DECLARE_IMAGE_MODE1	121
5.3.3 Typedef Documentation	121
5.3.3.1 TY_ACC_BIAS	121
5.3.3.2 TY_ACC_MISALIGNMENT	121
5.3.3.3 TY_ACC_SCALE	122
5.3.3.4 TY_ACCESS_MODE_LIST	122
5.3.3.5 TY_BYTEARRAY_ATTR	122
5.3.3.6 TY_CAMERA_CALIB_INFO	123

5.3.3.7 TY_CAMERA_DISTORTION	123
5.3.3.8 TY_CAMERA_EXTRINSIC	123
5.3.3.9 TY_CAMERA_INTRINSIC	123
5.3.3.10 TY_CAMERA_ROTATION	124
5.3.3.11 TY_CAMERA_TO_IMU	124
5.3.3.12 TY_COMPONENT_ID	124
5.3.3.13 TY_DEVICE_BASE_INFO	125
5.3.3.14 TY_DEVICE_COMPONENT_LIST	125
5.3.3.15 TY_ENUM_ENTRY	125
5.3.3.16 TY_FEATURE_ID	126
5.3.3.17 TY_FLOAT_RANGE	126
5.3.3.18 TY_GYRO_BIAS	126
5.3.3.19 TY_GYRO_MISALIGNMENT	126
5.3.3.20 TY_GYRO_SCALE	127
5.3.3.21 TY_INTERFACE_INFO	127
5.3.3.22 TY_INTERFACE_TYPE_LIST	127
5.3.3.23 TY_PIXEL_BITS_LIST	127
5.3.3.24 TY_TRIGGER_MODE_LIST	128
5.3.4 Enumeration Type Documentation	128
5.3.4.1 TY_ACCESS_MODE_LIST	128
5.3.4.2 TY_DEVICE_COMPONENT_LIST	128
5.3.4.3 TY_FEATURE_ID_LIST	129
5.3.4.4 TY_INTERFACE_TYPE_LIST	131
5.3.4.5 TY_PIXEL_BITS_LIST	132
5.3.4.6 TY_PIXEL_FORMAT_LIST	132
5.3.4.7 TY_RESOLUTION_MODE_LIST	133
5.3.4.8 TY_TRIGGER_MODE_LIST	133
5.4 TYImageProc.h File Reference	134
5.4.1 Detailed Description	135
5.4.2 Function Documentation	135
5.4.2.1 TYDepthEnhenceFilter()	135
5.4.2.2 TYDepthSpeckleFilter()	136
5.4.2.3 TYImageProcesAcceEnable()	136
5.4.2.4 TYUndistortImage()	136

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.

Chapter 2

Class Index

2.1 Class List

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

DepthEnhenceParameters	7
DepthSpeckleFilterParameters	7
gz_header_s	8
gzFile_s	8
pattern_bin_param	8
pattern_gray_param	9
pattern_sine_param	9
TY_PHC_GROUP_ATTR::phc_group_attr	10
TY_ACC_BIAS	10
TY_ACC_MISALIGNMENT	10
TY_ACC_SCALE	11
TY_AEC_ROI_PARAM	12
TY_BYTEARRAY_ATTR	
Byte array data structure	12
TY_CAMERA_CALIB_INFO	13
TY_CAMERA_DISTORTION	14
TY_CAMERA_EXTRINSIC	14
TY_CAMERA_INTRINSIC	15
TY_CAMERA_ROTATION	16
TY_CAMERA_STATISTICS	16
TY_CAMERA_TO_IMU	17
TY_DEVICE_BASE_INFO	17
TY_DEVICE_NET_INFO	
Device network information	18
TY_DEVICE_USB_INFO	19
TY_DI_WORKMODE	19
TY_DO_WORKMODE	20
TY_ENUM_ENTRY	20
TY_EVENT_INFO	21
TY_FEATURE_INFO	21
TY_FLOAT_RANGE	
Float range data structure	22
TY_FRAME_DATA	22
TY_GYRO_BIAS	23
TY_GYRO_MISALIGNMENT	23

TY_GYRO_SCALE	24
TY_IMAGE_DATA	25
TY_IMU_DATA	25
TY_INT_RANGE	26
TY_INTERFACE_INFO	26
TY_LASER_PARAM	27
TY_LASER_PATTERN_PARAM	27
TY_PHC_GROUP_ATTR	28
TY_PIXEL_COLOR_DESC	29
TY_PIXEL_DESC	29
TY_TEMP_DATA	30
TY_TOF_FREQ	30
TY_TRIGGER_PARAM	30
TY_TRIGGER_PARAM_EX	31
TY_TRIGGER_TIMER_LIST	31
TY_TRIGGER_TIMER_PERIOD	32
TY_VECT_3F	32
TY_VERSION_INFO	32
TYEnumEntry	33
z_stream_s	33

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	35
TYCoordinateMapper.h	Coordinate Conversion API	104
TYDefs.h	TYDefs.h includes camera control and data receiving data definitions which supports configuration for image resolution, frame rate, exposure time, gain, working mode,etc	110
TYImageProc.h	134
TYParameter.h	??
TYVer.h	??
zconf.h	??
zlib.h	??

Chapter 4

Class Documentation

4.1 DepthEnhanceParameters 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 gz_header_s Struct Reference

Public Attributes

- int **text**
- uLong **time**
- int **xflags**
- int **os**
- Bytef * **extra**
- uInt **extra_len**
- uInt **extra_max**
- Bytef * **name**
- uInt **name_max**
- Bytef * **comment**
- uInt **comm_max**
- int **hcrc**
- int **done**

4.3.1 Detailed Description

Definition at line 114 of file zlib.h.

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

- zlib.h

4.4 gzFile_s Struct Reference

Public Attributes

- unsigned **have**
- unsigned char * **next**
- z_off64_t **pos**

4.4.1 Detailed Description

Definition at line 1837 of file zlib.h.

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

- zlib.h

4.5 pattern_bin_param Struct Reference

Public Attributes

- uint32_t **offset**
- uint8_t **data** [512]

4.5.1 Detailed Description

Definition at line 1068 of file TYDefs.h.

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

- [TYDefs.h](#)

4.6 pattern_gray_param Struct Reference

Public Attributes

- uint32_t **phase_num**
- uint32_t **param1**
- uint32_t **param2**
- uint32_t **param3**

4.6.1 Detailed Description

Definition at line 1060 of file TYDefs.h.

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

- [TYDefs.h](#)

4.7 pattern_sine_param Struct Reference

Public Attributes

- uint32_t **phase_num**
- float **period**

4.7.1 Detailed Description

Definition at line 1054 of file TYDefs.h.

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

- [TYDefs.h](#)

4.8 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]

4.8.1 Detailed Description

Definition at line 1038 of file TYDefs.h.

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

- [TYDefs.h](#)

4.9 TY_ACC_BIAS Struct Reference

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

Public Attributes

- float **data** [3]

4.9.1 Detailed Description

a 3x3 matrix

.	.	.
BIASx	BIASy	BIASz

Definition at line 1118 of file TYDefs.h.

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

- [TYDefs.h](#)

4.10 TY_ACC_MISALIGNMENT Struct Reference

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

Public Attributes

- float **data** [3 *3]

4.10.1 Detailed Description

a 3x3 matrix

|.|.|

.	.	.
1	-GAM _{Ayz}	GAM _{Azy}
GAM _{Axz}	1	-GAM _{Azx}
-GAM _{Axy}	GAM _{Ayx}	1

Definition at line 1130 of file TYDefs.h.

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

- [TYDefs.h](#)

4.11 TY_ACC_SCALE Struct Reference

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

Public Attributes

- float **data** [3 *3]

4.11.1 Detailed Description

a 3x3 matrix

.	.	.
SCALE _x	0	0
0	SCALE _y	0
0	0	SCALE _z

Definition at line 1141 of file TYDefs.h.

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

- [TYDefs.h](#)

4.12 TY_AEC_ROI_PARAM Struct Reference

Public Attributes

- `uint32_t x`
- `uint32_t y`
- `uint32_t w`
- `uint32_t h`

4.12.1 Detailed Description

Definition at line 1018 of file TYDefs.h.

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

- [TYDefs.h](#)

4.13 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`

4.13.1 Detailed Description

byte array data structure

See also

[TYGetByteArray](#)

Definition at line 886 of file TYDefs.h.

4.13.2 Member Data Documentation

4.13.2.1 unit_size

```
int32_t TY_BYTEARRAY_ATTR::unit_size
```

unit size in bytes for special parse

Definition at line 889 of file TYDefs.h.

4.13.2.2 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 892 of file TYDefs.h.

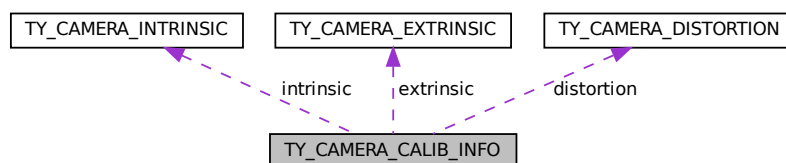
The documentation for this struct was generated from the following file:

- [TYDefs.h](#)

4.14 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`

4.14.1 Detailed Description

camera 's caillbration data

See also

[TYGetStruct](#)

Definition at line 961 of file TYDefs.h.

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

- [TYDefs.h](#)

4.15 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.

4.15.1 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 953 of file TYDefs.h.

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

- [TYDefs.h](#)

4.16 TY_CAMERA_EXTRINSIC Struct Reference

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

Public Attributes

- float [data](#) [4 *4]

4.16.1 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 941 of file TYDefs.h.

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

- [TYDefs.h](#)

4.17 TY_CAMERA_INTRINSIC Struct Reference

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

Public Attributes

- float **data** [3 *3]

4.17.1 Detailed Description

a 3x3 matrix

.	.	.
fx	0	cx
0	fy	cy
0	0	1

See also

[TYGetStruct](#) Usage:

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

Definition at line 923 of file TYDefs.h.

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

- [TYDefs.h](#)

4.18 TY_CAMERA_ROTATION Struct Reference

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

Public Attributes

- float **data** [3 *3]

4.18.1 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 1299 of file TYDefs.h.

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

- [TYDefs.h](#)

4.19 TY_CAMERA_STATISTICS Struct Reference

Public Attributes

- uint64_t **packetReceived**
- uint64_t **packetLost**
- uint64_t **imageOutputed**
- uint64_t **imageDropped**
- uint8_t **rsvd** [1024]

4.19.1 Detailed Description

Definition at line 1092 of file TYDefs.h.

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

- [TYDefs.h](#)

4.20 TY_CAMERA_TO_IMU Struct Reference

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

Public Attributes

- float **data** [4 *4]

4.20.1 Detailed Description

a 4x4 matrix

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

Definition at line 1184 of file TYDefs.h.

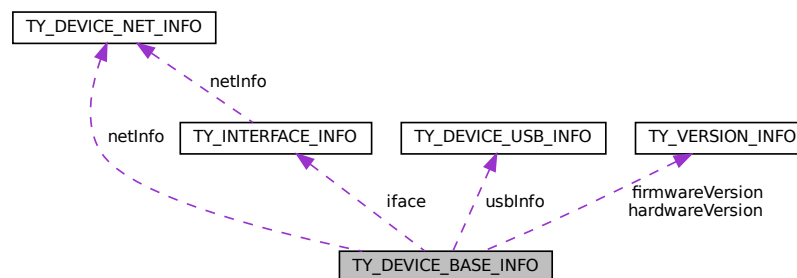
The documentation for this struct was generated from the following file:

- [TYDefs.h](#)

4.21 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]
- char **configVersion** [256]
- char **reserved** [256]

4.21.1 Detailed Description

See also

[TYGetDeviceList](#)

Definition at line 824 of file TYDefs.h.

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

- [TYDefs.h](#)

4.22 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 **tlversion** [32]
- char **reserved** [64]

4.22.1 Detailed Description

device network information

Definition at line 795 of file TYDefs.h.

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

- [TYDefs.h](#)

4.23 TY_DEVICE_USB_INFO Struct Reference

Public Attributes

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

4.23.1 Detailed Description

Definition at line 806 of file TYDefs.h.

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

- [TYDefs.h](#)

4.24 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]

4.24.1 Detailed Description

Definition at line 1266 of file TYDefs.h.

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

- [TYDefs.h](#)

4.25 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]

4.25.1 Detailed Description

Definition at line 1243 of file TYDefs.h.

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

- [TYDefs.h](#)

4.26 TY_ENUM_ENTRY Struct Reference

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

Public Attributes

- char **description** [64]
- uint32_t **value**
- uint32_t **reserved** [3]

4.26.1 Detailed Description

enum feature entry information

See also

[TYGetEnumEntryInfo](#)

Definition at line 897 of file TYDefs.h.

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

- [TYDefs.h](#)

4.27 TY_EVENT_INFO Struct Reference

Public Attributes

- TY_EVENT **eventId**
- char **message** [124]

4.27.1 Detailed Description

Definition at line 1237 of file TYDefs.h.

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

- [TYDefs.h](#)

4.28 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]

4.28.1 Detailed Description

Definition at line 850 of file TYDefs.h.

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

- [TYDefs.h](#)

4.29 TY_FLOAT_RANGE Struct Reference

float range data structure

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

Public Attributes

- float **min**
- float **max**
- float **inc**
increasing step
- float **reserved** [1]

4.29.1 Detailed Description

float range data structure

See also

[TYGetFloatRange](#)

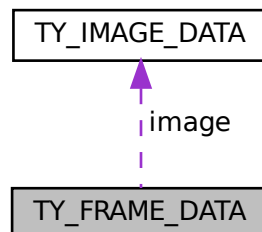
Definition at line 876 of file TYDefs.h.

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

- [TYDefs.h](#)

4.30 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.

4.30.1 Detailed Description

Definition at line 1227 of file TYDefs.h.

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

- [TYDefs.h](#)

4.31 TY_GYRO_BIAS Struct Reference

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

Public Attributes

- float [data](#) [3]

4.31.1 Detailed Description

a 3x3 matrix

.	.	.
BIASx	BIASy	BIASz

Definition at line 1150 of file TYDefs.h.

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

- [TYDefs.h](#)

4.32 TY_GYRO_MISALIGNMENT Struct Reference

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

Public Attributes

- float **data** [3 *3]

4.32.1 Detailed Description

a 3x3 matrix

.	.	.
1	-ALPHAy _z	ALPHA _z y
0	1	-ALPHA _z x
0	0	1

Definition at line 1161 of file TYDefs.h.

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

- [TYDefs.h](#)

4.33 TY_GYRO_SCALE Struct Reference

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

Public Attributes

- float **data** [3 *3]

4.33.1 Detailed Description

a 3x3 matrix

.	.	.
SCALE _x	0	0
0	SCALE _y	0
0	0	SCALE _z

Definition at line 1172 of file TYDefs.h.

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

- [TYDefs.h](#)

4.34 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.
- [TYPixFmt](#) [pixelFormat](#)
Pixel format, see TY_PIXEL_FORMAT_LIST.
- int32_t [reserved](#) [9]
Reserved.

4.34.1 Detailed Description

Definition at line 1212 of file TYDefs.h.

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

- [TYDefs.h](#)

4.35 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]

4.35.1 Detailed Description

Definition at line 1101 of file TYDefs.h.

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

- [TYDefs.h](#)

4.36 TY_INT_RANGE Struct Reference

Public Attributes

- `int32_t` **min**
- `int32_t` **max**
- `int32_t` **inc**
increasing step
- `int32_t` **reserved** [1]

4.36.1 Detailed Description

Definition at line 866 of file TYDefs.h.

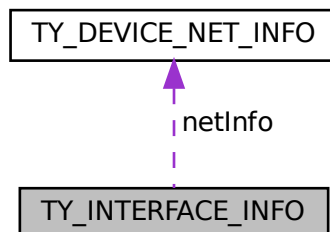
The documentation for this struct was generated from the following file:

- [TYDefs.h](#)

4.37 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**

4.37.1 Detailed Description

See also

[TYGetInterfaceList](#)

Definition at line 814 of file TYDefs.h.

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

- [TYDefs.h](#)

4.38 TY_LASER_PARAM Struct Reference

Public Attributes

- uint32_t **idx**
- uint32_t **en**
- uint32_t **power**

4.38.1 Detailed Description

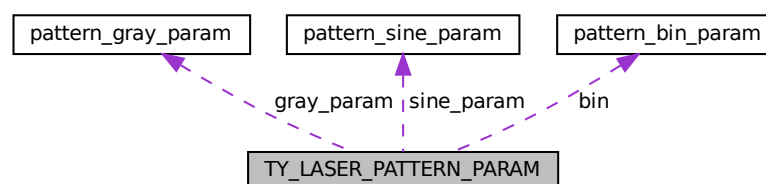
Definition at line 1202 of file TYDefs.h.

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

- [TYDefs.h](#)

4.39 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**
 };

4.39.1 Detailed Description

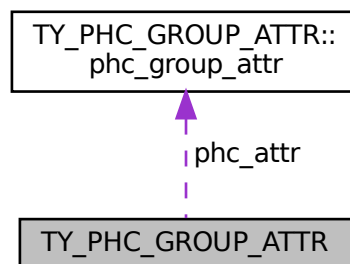
Definition at line 1074 of file TYDefs.h.

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

- [TYDefs.h](#)

4.40 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]

4.40.1 Detailed Description

Definition at line 1034 of file TYDefs.h.

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

- [TYDefs.h](#)

4.41 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`

4.41.1 Detailed Description

Definition at line 20 of file TYCoordinateMapper.h.

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

- [TYCoordinateMapper.h](#)

4.42 TY_PIXEL_DESC Struct Reference

Public Attributes

- `int16_t x`
- `int16_t y`
- `uint16_t depth`
- `uint16_t rsvd`

4.42.1 Detailed Description

Definition at line 12 of file TYCoordinateMapper.h.

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

- [TYCoordinateMapper.h](#)

4.43 TY_TEMP_DATA Struct Reference

Public Attributes

- uint32_t **id**
- char **name** [16]
- char **temp** [16]
- char **desc** [16]

4.43.1 Detailed Description

Definition at line 1280 of file TYDefs.h.

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

- [TYDefs.h](#)

4.44 TY_TOF_FREQ Struct Reference

Public Attributes

- uint32_t **freq1**
- uint32_t **freq2**

4.44.1 Detailed Description

Definition at line 1189 of file TYDefs.h.

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

- [TYDefs.h](#)

4.45 TY_TRIGGER_PARAM Struct Reference

Public Attributes

- TY_TRIGGER_MODE **mode**
- int8_t **fps**
- int8_t **rsvd**

4.45.1 Detailed Description

Definition at line 972 of file TYDefs.h.

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

- [TYDefs.h](#)

4.46 TY_TRIGGER_PARAM_EX Struct Reference

Public Attributes

- TY_TRIGGER_MODE **mode**
-

```
union {  
    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]  
};
```

4.46.1 Detailed Description

Definition at line 980 of file TYDefs.h.

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

- [TYDefs.h](#)

4.47 TY_TRIGGER_TIMER_LIST Struct Reference

Public Attributes

- uint64_t **start_time_us**
- uint32_t **offset_us_count**
- uint32_t **offset_us_list** [50]

4.47.1 Detailed Description

Definition at line 1003 of file TYDefs.h.

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

- [TYDefs.h](#)

4.48 TY_TRIGGER_TIMER_PERIOD Struct Reference

Public Attributes

- uint64_t **start_time_us**
- uint32_t **trigger_count**
- uint32_t **period_us**

4.48.1 Detailed Description

Definition at line 1011 of file TYDefs.h.

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

- [TYDefs.h](#)

4.49 TY_VECT_3F Struct Reference

Public Attributes

- float **x**
- float **y**
- float **z**

4.49.1 Detailed Description

Definition at line 904 of file TYDefs.h.

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

- [TYDefs.h](#)

4.50 TY_VERSION_INFO Struct Reference

Public Attributes

- int32_t **major**
- int32_t **minor**
- int32_t **patch**
- int32_t **reserved**

4.50.1 Detailed Description

Definition at line 786 of file TYDefs.h.

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

- [TYDefs.h](#)

4.51 TYEnumEntry Struct Reference

Public Attributes

- `int32_t` **value**
- `char` **name** [64]
- `char` **tooltip** [512]
- `char` **description** [512]
- `char` **displayName** [512]

4.51.1 Detailed Description

Definition at line 16 of file TYParameter.h.

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

- [TYParameter.h](#)

4.52 z_stream_s Struct Reference

Public Attributes

- `z_const Bytef *` **next_in**
- `uInt` **avail_in**
- `uLong` **total_in**
- `Bytef *` **next_out**
- `uInt` **avail_out**
- `uLong` **total_out**
- `z_const char *` **msg**
- `struct internal_state FAR *` **state**
- `alloc_func` **zalloc**
- `free_func` **zfree**
- `voidpf` **opaque**
- `int` **data_type**
- `uLong` **adler**
- `uLong` **reserved**

4.52.1 Detailed Description

Definition at line 86 of file zlib.h.

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

- [zlib.h](#)

Chapter 5

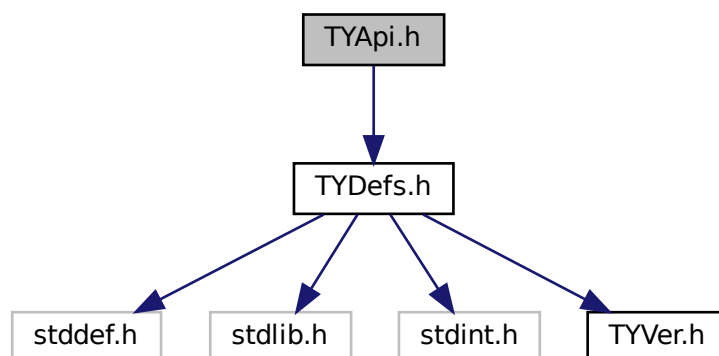
File Documentation

5.1 TYApi.h File Reference

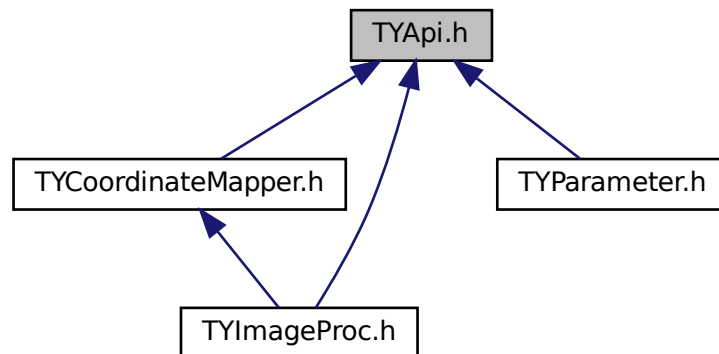
[TYApi.h](#) includes camera control and data receiving interface, which supports configuration for image resolution, frame rate, exposure time, gain, working mode, etc.

```
#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 *pIfaceInfos, uint32_t bufferCount, uint32_t *filledCount)
 - 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.*
- TY_CAPI [TYGetDeviceNumber](#) (TY_INTERFACE_HANDLE ifaceHandle, uint32_t *deviceNumber)
 - Get number of current connected devices.*
- TY_CAPI [TYGetDeviceList](#) (TY_INTERFACE_HANDLE ifaceHandle, TY_DEVICE_BASE_INFO *deviceInfos, 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 *pIface)
 - 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.*
- TY_CAPI [TYGetDeviceInfo](#) (TY_DEV_HANDLE hDevice, TY_DEVICE_BASE_INFO *info)
 - Get base info of the open device.*
- 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()

```
TY_CAPI TYAppendLogToFile (
    const char * filePath,
    TY_LOG_LEVEL lvl )
```

Append log to specified file.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_ERROR</i>	Failed to add file Suggestions: Please check if the file path is correct and if you have permission to write to

5.1.2.2 TYAppendLogToServer()

```
TY_CAPI TYAppendLogToServer (
    const char * protocol,
    const char * ip,
    uint16_t port,
    TY_LOG_LEVEL lvl )
```

Append log to Tcp/Udp server.

Parameters

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

Return values

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

5.1.2.3 TYClearBufferQueue()

```
TY_CAPI TYClearBufferQueue (
    TY_DEV_HANDLE hDevice )
```

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

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYClearBufferQueue called with invalid device handle</p> <p>Suggestions:</p> <p>Please check device handle</p> <p>Like this:</p> <pre>TYClearBufferQueue(hDevice); ^ is invalid</pre> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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
<i>TY_STATUS_BUSY</i>	Device is capturing.

5.1.2.4 TYCloseDevice()

```
TY_CAPI TYCloseDevice (
    TY_DEV_HANDLE hDevice,
    bool reboot = false )
```

Close device by device handle.

Parameters

in	<i>hDevice</i>	Device handle.
in	<i>reboot</i>	Reboot device after close.

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

Return values

<i>TY_STATUS_INVALID_HANDLE</i>	TYCloseDevice called with invalid device handle Suggestions: 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 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUpdated
<i>TY_STATUS_TIMEOUT</i>	Failed to close device Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network
<i>TY_STATUS_DEVICE_ERROR</i>	Failed to close device Suggestions: Possible reasons: 1.Camera device is abnormal and cannot be closed normally.
<i>TY_STATUS_IDLE</i>	Device has been closed.

5.1.2.5 TYCloseInterface()

```
TY_CAPI TYCloseInterface (
    TY_INTERFACE_HANDLE ifaceHandle )
```

Close interface.

Parameters

in	<i>ifaceHandle</i>	Interface to be closed.
----	--------------------	-------------------------

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	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 correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling TYU

5.1.2.6 TYDeinitLib()

```
TY_CAPI TYDeinitLib (  
    void )
```

Deinit this library.

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

5.1.2.7 TYDisableComponents()

```
TY_CAPI TYDisableComponents (  
    TY_DEV_HANDLE hDevice,  
    TY_COMPONENT_ID componentIDs )
```

Disable components.

Parameters

in	<i>hDevice</i>	Device handle.
in	<i>componentIDs</i>	Components to be disabled.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYDisableComponents called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: TYDisableComponents(hDevice, componentIDs); ^ is invalid</p> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none">1.TYOpenDevice failed to open device and get correct handle2.Memory in stack to store handle data is corrupted3.After getting handle, you updated device list by calling TYUp
<i>TY_STATUS_INVALID_PARAMETER</i>	<p>Invalid component IDs</p> <p>Suggestions: Please check componentIDs parameter Like this: TYDisableComponents(hDevice, componentIDs); ^ is invalid</p> <p>componentIDs should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	Some components specified by componentIDs are invalid.

Return values

<i>TY_STATUS_BUSY</i>	Camera device is capturing Suggestions: Please call <code>TYEnableComponents</code> when the camera device is stopped Like this: <pre> TYStopCapture(hDevice); TYDisableComponents(hDevice, componentIDs); </pre>
-----------------------	---

See also

[TY_DEVICE_COMPONENT_LIST](#)

5.1.2.8 TYEnableComponents()

```

TY_CAPI TYEnableComponents (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentIDs )

```

Enable components.

Parameters

in	<i>hDevice</i>	Device handle.
in	<i>componentIDs</i>	Components to be enabled.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	TYEnableComponents called with invalid device handle Suggestions: Please check device handle Like this: <pre> TYEnableComponents(hDevice, componentIDs); ^ is invalid </pre> The <code>hDevice</code> 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
<i>TY_STATUS_INVALID_PARAMETER</i>	Invalid component IDs Suggestions: Please check <code>componentIDs</code> parameter Like this: <pre> TYEnableComponents(hDevice, componentIDs); ^ is invalid </pre> <code>componentIDs</code> should be the value returned by <code>TYGetComponentIDs</code> You can also view the components of the camera by obtaining the x
<i>TY_STATUS_INVALID_COMPONENT</i>	Some components specified by <code>componentIDs</code> are invalid.

Return values

<i>TY_STATUS_BUSY</i>	Camera device is capturing Suggestions: Please call <code>TYEnableComponents</code> when the camera device is stopped Like this: <pre>TYStopCapture(hDevice); TYEnableComponents(hDevice, componentIDs);</pre>
-----------------------	--

See also

[TY_DEVICE_COMPONENT_LIST](#)

5.1.2.9 TYEnqueueBuffer()

```
TY_CAPI TYEnqueueBuffer (  
    TY_DEV_HANDLE hDevice,  
    void * buffer,  
    uint32_t bufferSize )
```

Enqueue a user allocated buffer.

Parameters

in	<i>hDevice</i>	Device handle.
in	<i>buffer</i>	Buffer to be enqueued.
in	<i>bufferSize</i>	Size of the input buffer.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	TYEnqueueBuffer called with invalid device handle Suggestions: Please check device handle Like this: <pre>TYEnqueueBuffer(hDevice, buffer, bufferSize); ^ is invalid</pre> The <code>hDevice</code> 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 <code>TYUpdated</code>
<i>TY_STATUS_NULL_POINTER</i>	TYEnqueueBuffer called with NULL pointer Suggestions: Please check your code Like this: <pre>TYEnqueueBuffer(hDevice, buffer, bufferSize); ^ is NULL</pre>

Return values

<i>TY_STATUS_WRONG_SIZE</i>	TYEnqueueBuffer called with wrong size Suggestions: Please check your code Like this: <pre>TYEnqueueBuffer(hDevice, buffer, bufferSize);</pre> ^ is 0 or negative value
<i>TY_STATUS_TIMEOUT</i>	Failed to enqueue frame buffer Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network
<i>TY_STATUS_DEVICE_ERROR</i>	Failed to enqueue frame buffer Suggestions: Possible reasons: 1.Camera device is abnormal and cannot get the frame buffer size.

5.1.2.10 TYErrorString()

```
TY_EXTC const TY_EXPORT char* TY_STDC TYErrorString (
    TY_STATUS errorID )
```

Get error information.

Parameters

in	<i>errorID</i>	Error id.
----	----------------	-----------

Return values

<i>Error</i>	string.
--------------	---------

5.1.2.11 TYFetchFrame()

```
TY_CAPI TYFetchFrame (
    TY_DEV_HANDLE hDevice,
    TY_FRAME_DATA * frame,
    int32_t timeout )
```

Fetch one frame.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYFetchFrame called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <pre>TYFetchFrame(hDevice, pFrame, timeout);</pre> ^ is invalid</p> <p>The hDevice parameter you input is not recorded</p> <p>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</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYFetchFrame called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <pre>TYFetchFrame(hDevice, pFrame, timeout);</pre> ^ is NULL</p>
<i>TY_STATUS_IDLE</i>	<p>Camera device is not started</p> <p>Suggestions: Please start the camera device first Like this: <pre>TYStartCapture(hDevice);</pre> <pre>TYFetchFrame(hDevice, pFrame, timeout);</pre></p>
<i>TY_STATUS_WRONG_MODE</i>	Callback has been registered, this function is disabled.
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to get frame</p> <p>Suggestions: Possible reasons: 1.Camera device is abnormal and cannot get frame. 2.Network communication is abnormal, please check whether the network 3.Timeout, frame acquisition timeout</p>

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

Return values

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

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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
out	<i>value</i>	Bool value.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetBool called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <pre>TYGetBool(hDevice, componentID, featureID, pValue);</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 calling TYUpd</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <pre>TYGetBool(hDevice, componentID, featureID, pValue);</pre> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <pre>TYGetBool(hDevice, componentID, featureID, pValue);</pre> ^ 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 t</p>
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <pre>TYGetBool(hDevice, componentID, featureID, pValue);</pre> ^ type mismatch The feature type you entered does not match. You can use TYFeatur</p>

Return values

<i>TY_STATUS_NULL_POINTER</i>	TYGetBool called with NULL pointer Suggestions: Please check your code Like this: <pre>TYGetBool(hDevice, componentID, featureID, pValue); ^ is NULL</pre>
<i>TY_STATUS_TIMEOUT</i>	Failed to get bool feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network is normal
<i>TY_STATUS_DEVICE_ERROR</i>	Failed to get bool feature Suggestions: Possible reasons: 1.The feature of the camera device is not available or not implemented 2.Camera device is abnormal and cannot get bool feature

5.1.2.14 TYGetByteArray()

```

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.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	TYGetByteArray called with invalid device handle Suggestions: Please check device handle Like this: <pre>TYGetByteArray(hDevice, componentID, featureID, pBuffer, bufferSize); ^ is invalid</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 TYUpdateDeviceList

Return values

<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions:</p> <p>Please check componentID parameter</p> <p>Like this:</p> <pre>TYGetByteArray(hDevice, componentID, featureID, buffer, bufferSize);</pre> <p>^ is invalid</p> <p>componentID should be the value returned by TYGetComponentIDs</p> <p>You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions:</p> <p>Please check featureID parameter</p> <p>Like this:</p> <pre>TYGetByteArray(hDevice, componentID, featureID, buffer, bufferSize);</pre> <p>^ is invalid</p> <p>You entered an invalid featureID parameter</p> <p>You can get a list of features of the camera device through TYGet</p> <p>You can also view the features of the camera device by obtaining t</p>
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions:</p> <p>Please check the feature type</p> <p>Like this:</p> <pre>TYGetByteArray(hDevice, componentID, featureID, buffer, bufferSize);</pre> <p>^ type mismatch</p> <p>The feature type you entered does not match. You can use TYFeature</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetByteArray called with NULL pointer</p> <p>Suggestions:</p> <p>Please check your code</p> <p>Like this:</p> <pre>TYGetByteArray(hDevice, componentID, featureID, pBuffer, bufferSize);</pre> <p>^ is NULL</p>
<i>TY_STATUS_WRONG_SIZE</i>	<p>Array size mismatch</p> <p>Suggestions:</p> <p>Please check the array size</p> <p>Like this:</p> <pre>TYGetByteArray(hDevice, componentID, featureID, buffer, bufferSize);</pre> <p>^ is inv</p> <p>The array size you entered does not match</p>
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to get byte array feature</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.Network communication is abnormal, please check whether the ne
<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to get byte array feature</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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_CAPI TYGetByteArrayAttr (
    TY_DEV_HANDLE hDevice,
```

```

TY_COMPONENT_ID componentID,
TY_FEATURE_ID featureID,
TY_BYTEARRAY_ATTR * pAttr )

```

Write byte array to device.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetByteArrayAttr called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <pre>TYGetByteArrayAttr(hDevice, componentID, featureID, pAttr);</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 calling TYUpd</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <pre>TYGetByteArrayAttr(hDevice, componentID, featureID, pAttr);</pre> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <pre>TYGetByteArrayAttr(hDevice, componentID, featureID, pAttr);</pre> ^ 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 t</p>
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <pre>TYGetByteArrayAttr(hDevice, componentID, featureID, pAttr);</pre> ^ type mismatch The feature type you entered does not match. You can use TYFeature</p>

Return values

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	TYGetDeviceFeatureInfo called with invalid device handle Suggestions: Please check device handle Like this: TYGetDeviceFeatureInfo(hDevice, componentID, featureInfo, entryCount, filledEntryCount); ^ 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 TYUpdateDeviceList

Return values

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

5.1.2.19 TYGetDeviceFeatureNumber()

```

TY_CAPI TYGetDeviceFeatureNumber (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    uint32_t * size )

```

Get the size of device features.

Parameters

in	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
out	<i>size</i>	Size of all feature cnt.

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

Return values

<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetDeviceFeatureNumber called with invalid device handle</p> <p>Suggestions:</p> <p>Please check device handle</p> <p>Like this:</p> <pre>TYGetDeviceFeatureNumber(hDevice, componentID, size); ^ is invalid</pre> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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 TYUpdateDeviceList
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions:</p> <p>Please check componentID parameter</p> <p>Like this:</p> <pre>TYGetDeviceFeatureNumber(hDevice, componentID, size); ^ is invalid</pre> <p>componentID should be the value returned by TYGetComponentIDs</p> <p>You can also view the components of the camera by obtaining the xrt::device object</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetDeviceFeatureNumber called with NULL pointer</p> <p>Suggestions:</p> <p>Please check your code</p> <p>Like this:</p> <pre>TYGetDeviceFeatureNumber(hDevice, componentID, size); ^ is NULL</pre>
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to get feature number</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.Network communication is abnormal, please check whether the network is connected
<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to get feature number</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.The feature of the camera device is not available or not implemented 2.Camera device is abnormal and cannot get feature number

5.1.2.20 TYGetDeviceInfo()

```
TY_CAPI TYGetDeviceInfo (
    TY_DEV_HANDLE hDevice,
    TY_DEVICE_BASE_INFO * info )
```

Get base info of the open device.

Parameters

in	<i>hDevice</i>	Device handle.
out	<i>info</i>	Base info out.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetDeviceInfo called with invalid device handle</p> <p>Suggestions:</p> <p>Please check device handle</p> <p>Like this:</p> <pre>TYGetDeviceInfo(hDevice, info); ^ is invalid</pre> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetDeviceInfo called with NULL pointer</p> <p>Suggestions:</p> <p>Please check your code</p> <p>Like this:</p> <pre>TYGetDeviceInfo(hDevice, info); ^ is NULL</pre>

5.1.2.21 TYGetDeviceInterface()

```
TY_CAPI TYGetDeviceInterface (
    TY_DEV_HANDLE hDevice,
    TY_INTERFACE_HANDLE * pIface )
```

Get interface handle by device handle.

Parameters

in	<i>hDevice</i>	Device handle.
out	<i>pIface</i>	Interface handle.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetDeviceInterface called with invalid device handle</p> <p>Suggestions:</p> <p>Please check device handle</p> <p>Like this:</p> <pre>TYGetDeviceInterface(hDevice, pIface); ^ is invalid</pre> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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

Return values

<i>TY_STATUS_NULL_POINTER</i>	TYGetDeviceInterface called with NULL pointer Suggestions: Please check your code Like this: <pre>TYGetDeviceInterface(hDevice, pIface); ^ is NULL</pre>
--------------------------------------	--

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

Return values

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

5.1.2.23 TYGetDeviceNumber()

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

Get number of current connected devices.

Parameters

in	<i>ifaceHandle</i>	Interface handle.
out	<i>deviceNumber</i>	Number of connected devices.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	<p>TYGetDeviceNumber called with invalid interface handle</p> <p>Suggestions: Please check interface handle Like this: <code>TYGetDeviceNumber(ifaceHandle, pDeviceNumber);</code> ^ is invalid</p> <p>The ifaceHandle parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.TYOpenInterface failed to open interface and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling TYU
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetDeviceNumber called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <code>TYGetDeviceNumber(ifaceHandle, deviceNumber);</code> ^ is NULL</p>

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	Not call TYInitLib
<i>TY_STATUS_WRONG_SIZE</i>	<p>XML buffer size is not enough</p> <p>Suggestions: XML buffer size is not enough Like this: <code>TYGetDeviceXML(hDevice, xml, in_size, out_size);</code> <code>^</code> is invalid XML buffer size is not enough, please use TYGetDeviceXMLSize to get the size</p>
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetDeviceXML called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <code>TYGetDeviceXML(hDevice, xml, in_size, out_size);</code> <code>^</code> 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 TYUpdatedDeviceList</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetDeviceXML called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <code>TYGetDeviceXML(hDevice, xml, in_size, out_size);</code> <code>^</code> or <code>^</code> is NULL</p>

5.1.2.25 TYGetDeviceXMLSize()

```

TY_CAPI TYGetDeviceXMLSize (
    TY_DEV_HANDLE hDevice,
    uint32_t * size )

```

Get the Device xml size.

Parameters

in	<i>hDevice</i>	Device handle.
out	<i>size</i>	The size of device xml string

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	Not call TYInitLib

See also

[TY_DEVICE_COMPONENT_LIST](#)

5.1.2.27 TYGetEnum()

```
TY_CAPI TYGetEnum (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    uint32_t * value )
```

Get current value of enum feature.

Parameters

in	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
out	<i>value</i>	Enum value.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetEnum called with invalid device handle</p> <p>Suggestions:</p> <p>Please check device handle</p> <p>Like this:</p> <pre>TYGetEnum(hDevice, componentID, featureID, pValue);</pre> <p>^ is invalid</p> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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 TYUpd
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID: d</p> <p>Suggestions:</p> <p>Please check componentID parameter</p> <p>Like this:</p> <pre>TYGetEnum(hDevice, componentID, featureID, pValue);</pre> <p>^ is invalid</p> <p>componentID should be the value returned by TYGetComponentIDs</p> <p>You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID: d</p> <p>Suggestions:</p> <p>Please check featureID parameter</p> <p>Like this:</p> <pre>TYGetEnum(hDevice, componentID, featureID, pValue);</pre> <p>^ is invalid</p> <p>You entered an invalid featureID parameter</p> <p>You can get a list of features of the camera device through TYGetI</p> <p>You can also view the features of the camera device by obtaining t</p>

Return values

TY_STATUS_INVALID_HANDLE	<p>TYGetEnumEntryCount called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <pre>TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount, ^)</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 calling TYUpdateDeviceList</p>
TY_STATUS_INVALID_COMPONENT	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <pre>TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount, ^)</pre> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the xrt::device::component_id_list</p>
TY_STATUS_INVALID_FEATURE	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <pre>TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount, ^)</pre> ^ is invalid You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetFeatureIDs You can also view the features of the camera device by obtaining the xrt::device::feature_id_list</p>
TY_STATUS_WRONG_TYPE	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <pre>TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount, ^)</pre> ^ type mismatch The feature type you entered does not match. You can use TYFeatureType to get the feature type</p>
TY_STATUS_NULL_POINTER	<p>TYGetEnumEntryCount called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <pre>TYGetEnumEntryCount(hDevice, componentID, featureID, pEntryCount, ^)</pre> ^ is NULL</p>

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetEnumEntryInfo called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <code>TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, entryCount)</code> <code>^</code> is invalid</p> <p>The hDevice parameter you input is not recorded</p> <p>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 TYUpdateDeviceList</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID: d</p> <p>Suggestions: Please check componentID parameter Like this: <code>TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, entryCount)</code> <code>^</code> is invalid</p> <p>componentID should be the value returned by TYGetComponentIDs</p> <p>You can also view the components of the camera by obtaining the xrt::device object</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID: d</p> <p>Suggestions: Please check featureID parameter Like this: <code>TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, entryCount)</code> <code>^</code> is invalid</p> <p>You entered an invalid featureID parameter</p> <p>You can get a list of features of the camera device through TYGetFeatureIDs</p> <p>You can also view the features of the camera device by obtaining the xrt::device object</p>
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <code>TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, entryCount)</code> <code>^</code> type mismatch</p> <p>The feature type you entered does not match. You can use TYFeatureType to get the feature type</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetEnumEntryInfo called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <code>TYGetEnumEntryInfo(hDevice, componentID, featureID, pEnumDescriptions, entryCount)</code> <code>^</code></p>

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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
out	<i>featureInfo</i>	Feature info.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetFeatureInfo called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <code>TYGetFeatureInfo(hDevice, componentID, featureID, pFeatureInfo),</code> <code>^</code> 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 TYUpd</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <code>TYGetFeatureInfo(hDevice, componentID, featureID, pFeatureInfo),</code> <code>^</code> is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <code>TYGetFeatureInfo(hDevice, componentID, featureID, pFeatureInfo),</code> <code>^</code> is invalid You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetF You can also view the features of the camera device by obtaining t</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetFeatureInfo called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <code>TYGetFeatureInfo(hDevice, componentID, featureID, pFeatureInfo),</code> <code>^</code> is NULL</p>

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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
out	<i>value</i>	Float value.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetFloat called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <code>TYGetFloat(hDevice, componentID, featureID, pValue);</code> ^ 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 TYUpd</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <code>TYGetFloat(hDevice, componentID, featureID, pValue);</code> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <code>TYGetFloat(hDevice, componentID, featureID, pValue);</code> ^ is invalid You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetF You can also view the features of the camera device by obtaining t</p>
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <code>TYGetFloat(hDevice, componentID, featureID, pValue);</code> ^ type mismatch The feature type you entered does not match. You can use TYFeatur</p>

Return values

<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetFloat called with NULL pointer</p> <p>Suggestions:</p> <p>Please check your code</p> <p>Like this:</p> <pre>TYGetFloat(hDevice, componentID, featureID, pValue); // ^ is NULL</pre>
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to get float feature</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.Network communication is abnormal, please check whether the network is normal
<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to get float feature</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.The feature of the camera device is not available or not implemented 2.Camera device is abnormal and cannot get float feature

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetFloatRange called with invalid device handle</p> <p>Suggestions:</p> <p> Please check device handle</p> <p> Like this:</p> <pre> TYGetFloatRange(hDevice, componentID, featureID, pFloatRange); ^ is invalid </pre> <p> The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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 TYUpd

Return values

<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetFrameBufferSize called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <code>TYGetFrameBufferSize(hDevice, outSize);</code> <code>^</code> is invalid</p> <p>The hDevice parameter you input is not recorded</p> <p>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</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetFrameBufferSize called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <code>TYGetFrameBufferSize(hDevice, outSize);</code> <code>^</code> is NULL</p>
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to get frame buffer size</p> <p>Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network</p>
<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to get frame buffer size</p> <p>Suggestions: Possible reasons: 1.Camera device is abnormal and cannot get the frame buffer size.</p>

5.1.2.34 TYGetInt()

```

TY_CAPI TYGetInt (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    int32_t * value )

```

Get value of integer feature.

Parameters

in	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
out	<i>value</i>	Integer value.

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

Return values

<i>TY_STATUS_INVALID_HANDLE</i>	TYGetInt called with invalid device handle Suggestions: Please check device handle Like this: <pre>TYGetInt(hDevice, componentID, featureID, pValue);</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 calling TYUp
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID Suggestions: Please check componentID parameter Like this: <pre>TYGetInt(hDevice, componentID, featureID, pValue);</pre> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the xn
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID Suggestions: Please check featureID parameter Like this: <pre>TYGetInt(hDevice, componentID, featureID, pValue);</pre> ^ 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
<i>TY_STATUS_WRONG_TYPE</i>	Feature type mismatch Suggestions: Please check the feature type Like this: <pre>TYGetInt(hDevice, componentID, featureID, pValue);</pre> ^ type mismatch The feature type you entered does not match. You can use TYFeature
<i>TY_STATUS_NULL_POINTER</i>	TYGetInt called with NULL pointer Suggestions: Please check your code Like this: <pre>TYGetInt(hDevice, componentID, featureID, pValue);</pre> ^ is NULL
<i>TY_STATUS_TIMEOUT</i>	Failed to get int feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the n
<i>TY_STATUS_DEVICE_ERROR</i>	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()

```

TY_CAPI TYGetInterfaceList (
    TY_INTERFACE_INFO * pIfaceInfos,
    uint32_t bufferSize,
    uint32_t * filledCount )

```

Get interface info list.

Parameters

out	<i>pIfaceInfos</i>	Array of interface infos to be filled.
in	<i>bufferCount</i>	Array size of interface infos.
out	<i>filledCount</i>	Number of filled TY_INTERFACE_INFO .

Return values

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

5.1.2.36 TYGetInterfaceNumber()

```

TY_CAPI TYGetInterfaceNumber (
    uint32_t * pNumIfaces )

```

Get number of current interfaces.

Parameters

out	<i>pNumIfaces</i>	Number of interfaces.
-----	-------------------	-----------------------

Return values

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

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetIntRange called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <pre>TYGetIntRange(hDevice, componentID, featureID, pIntRange);</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 calling TYUpd</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <pre>TYGetIntRange(hDevice, componentID, featureID, pIntRange);</pre> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <pre>TYGetIntRange(hDevice, componentID, featureID, pIntRange);</pre> ^ 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 t</p>

Return values

<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions:</p> <p>Please check the feature type</p> <p>Like this:</p> <pre>TYGetIntRange(hDevice, componentID, featureID, pintRange);</pre> <p>^ type mismatch</p> <p>The feature type you entered does not match. You can use TYFeature</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetIntRange called with NULL pointer</p> <p>Suggestions:</p> <p>Please check your code</p> <p>Like this:</p> <pre>TYGetIntRange(hDevice, componentID, featureID, pintRange);</pre> <p>^ is NULL</p>

5.1.2.38 TYGetString()

```

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.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetString called with invalid device handle</p> <p>Suggestions:</p> <p>Please check device handle</p> <p>Like this:</p> <pre>TYGetString(hDevice, componentID, featureID, buffer, bufferSize);</pre> <p>^ is invalid</p> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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

Return values

<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions:</p> <p>Please check componentID parameter</p> <p>Like this:</p> <pre>TYGetString(hDevice, componentID, featureID, buffer, bufferSize)</pre> <p>^ is invalid</p> <p>componentID should be the value returned by TYGetComponentIDs</p> <p>You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions:</p> <p>Please check featureID parameter</p> <p>Like this:</p> <pre>TYGetString(hDevice, componentID, featureID, buffer, bufferSize)</pre> <p>^ is invalid</p> <p>You entered an invalid featureID parameter</p> <p>You can get a list of features of the camera device through TYGet</p> <p>You can also view the features of the camera device by obtaining t</p>
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions:</p> <p>Please check the feature type</p> <p>Like this:</p> <pre>TYGetString(hDevice, componentID, featureID, buffer, bufferSize)</pre> <p>^ type mismatch</p> <p>The feature type you entered does not match. You can use TYFeature</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetString called with NULL pointer</p> <p>Suggestions:</p> <p>Please check your code</p> <p>Like this:</p> <pre>TYGetString(hDevice, componentID, featureID, pBuffer, bufferSize)</pre> <p>^ is NULL</p>
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to get string feature</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.Network communication is abnormal, please check whether the n
<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to get string feature</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.The feature of the camera device is not available or not impl 2.Camera device is abnormal and cannot get string feature

See also

[TYGetStringLength](#)

5.1.2.39 TYGetStringLength()

```
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.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetString called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <code>TYGetString(hDevice, componentID, featureID, buffer, bufferSize);</code> <code>^</code> is invalid</p> <p>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 TYUpd</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <code>TYGetString(hDevice, componentID, featureID, buffer, bufferSize);</code> <code>^</code> is invalid</p> <p>componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <code>TYGetString(hDevice, componentID, featureID, buffer, bufferSize);</code> <code>^</code> is invalid</p> <p>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</p>
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <code>TYGetString(hDevice, componentID, featureID, buffer, bufferSize);</code> <code>^</code> type mismatch</p> <p>The feature type you entered does not match. You can use TYFeature</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYGetStringLength called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <code>TYGetStringLength(hDevice, componentID, featureID, pLength);</code> <code>^</code> is NULL</p>

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYGetStruct called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <pre>TYGetStruct(hDevice, componentID, featureID, pStruct, structSize)</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 calling TYUpd</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <pre>TYGetStruct(hDevice, componentID, featureID, pStruct, structSize)</pre> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <pre>TYGetStruct(hDevice, componentID, featureID, pStruct, structSize)</pre> ^ 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</p>
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <pre>TYGetStruct(hDevice, componentID, featureID, pStruct, structSize)</pre> ^ type mismatch The feature type you entered does not match. You can use TYFeature</p>

Return values

<i>TY_STATUS_NULL_POINTER</i>	TYGetStruct called with NULL pointer Suggestions: Please check your code Like this: TYGetStruct(hDevice, componentID, featureID, pStruct, structSize, ^ is NULL)
<i>TY_STATUS_WRONG_SIZE</i>	Struct size mismatch Suggestions: Please check the struct size Like this: TYGetStruct(hDevice, componentID, featureID, pStruct, structSize, ^ is invalid) The struct size you entered does not match
<i>TY_STATUS_TIMEOUT</i>	Failed to get struct feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network is normal
<i>TY_STATUS_DEVICE_ERROR</i>	Failed to get struct feature Suggestions: Possible reasons: 1.The feature of the camera device is not available or not implemented 2.Camera device is abnormal and cannot get struct feature

5.1.2.41 TYHasDevice()

```

TY_CAPI TYHasDevice (
    TY_INTERFACE_HANDLE ifaceHandle,
    const char * deviceID,
    bool * value )

```

Check whether the interface has the specified device.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.

Return values

<i>TY_STATUS_INVALID_INTERFACE</i>	<p>TYHasDevice called with invalid interface handle</p> <p>Suggestions: Please check interface handle Like this: <pre>TYHasDevice(ifaceHandle, deviceID, value);</pre> ^ is invalid</p> <p>The ifaceHandle parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.TYOpenInterface failed to open interface and get correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling TYU
<i>TY_STATUS_NULL_POINTER</i>	<p>TYHasDevice called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <pre>TYHasDevice(ifaceHandle, deviceID, value);</pre> ^ or ^ is NULL</p>

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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
out	<i>value</i>	Whether has feature.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYHasFeature called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <pre>TYHasFeature(hDevice, componentID, featureID, value);</pre> ^ is invalid</p> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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

5.1.2.44 TYLibVersion()

```
TY_CAPI TYLibVersion (
    TY_VERSION_INFO * version )
```

Get current library version.

Parameters

out	<i>version</i>	Version information to be filled.
-----	----------------	-----------------------------------

Return values

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

5.1.2.45 TYOpenDevice()

```
TY_CAPI TYOpenDevice (
    TY_INTERFACE_HANDLE ifaceHandle,
    const char * deviceID,
    TY_DEV_HANDLE * outDeviceHandle,
    TY_FW_ERRORCODE * outFwErrorcode = NULL )
```

Open device by device ID.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.

Return values

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

5.1.2.46 TYOpenDeviceWithIP()

```

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.

Parameters

in	<i>ifaceHandle</i>	Interface handle.
in	<i>IP</i>	Device IP.
out	<i>deviceHandle</i>	Handle of opened device.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	<p>TYOpenDeviceWithIP called with invalid interface handle</p> <p>Suggestions: Please check interface handle Like this: <code>TYOpenDeviceWithIP(ifaceHandle, IP, outDeviceHandle);</code> <code>^</code> is invalid</p> <p>The ifaceHandle parameter you input is not recorded</p> <p>Possible reasons: 1. TYOpenInterface failed to open interface and get correct handle 2. Memory in stack to store handle data is corrupted 3. After getting handle, you updated interface list by calling TYUpdateDeviceList</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYOpenDeviceWithIP called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <code>TYOpenDeviceWithIP(ifaceHandle, IP, outDeviceHandle);</code> <code>^</code> or <code>^</code> is NULL</p>
<i>TY_STATUS_INVALID_PARAMETER</i>	<p>TYOpenDeviceWithIP called with invalid IP address</p> <p>Suggestions: Please check your code Like this: <code>TYOpenDeviceWithIP(ifaceHandle, IP, outDeviceHandle);</code> <code>^</code> is invalid</p> <p>A valid IP address should be like: 192.168.31.1</p> <p>Usually you get device information by calling TYUpdateDeviceList, and then open device by calling TYOpenDevice</p> <p>When your device online status changes, you may need to update device list again</p>
<i>TY_STATUS_BUSY</i>	<p>Failed to open device</p> <p>Suggestions: Possible reasons: 1. Camera is occupied, please check if other processes on this machine or other host machines are occupying the camera. If the camera is occupied, please wait for a while and then try again. 2. A third-party program is written into the camera, please contact the camera manufacturer for more information.</p>

Return values

<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to open device</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.A third-party program is written into the camera, please contact the camera manufacturer. 2.The camera IP address cannot communicate with the host IP address. 3.Network communication is abnormal, please check whether the network is normal.
--------------------------------------	---

5.1.2.47 TYOpenInterface()

```

TY_CAPI TYOpenInterface (
    const char * ifaceID,
    TY_INTERFACE_HANDLE * outHandle )

```

Open specified interface.

Parameters

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

Return values

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

See also

[TYGetInterfaceList](#)

5.1.2.48 TYRegisterEventCallback()

```
TY_CAPI TYRegisterEventCallback (
    TY_DEV_HANDLE hDevice,
    TY_EVENT_CALLBACK callback,
    void * userdata )
```

Register device status callback. Register NULL to clean callback.

Parameters

in	<i>hDevice</i>	Device handle.
in	<i>callback</i>	Callback function.
in	<i>userdata</i>	User private data.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYRegisterEventCallback called with invalid device handle</p> <p>Suggestions:</p> <p>Please check device handle</p> <p>Like this:</p> <pre>TYRegisterEventCallback(hDevice, callback, userdata);</pre> <p>^ is invalid</p> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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
<i>TY_STATUS_BUSY</i>	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	<i>hDevice</i>	Device handle.
in	<i>callback</i>	Callback function.
in	<i>userdata</i>	User private data.

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

Return values

<i>TY_STATUS_INVALID_HANDLE</i>	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 calling TYUpdated
<i>TY_STATUS_BUSY</i>	Device is capturing.

5.1.2.50 TYRemoveLogFile()

```
TY_CAPI TYRemoveLogFile (
    const char * filePath )
```

Remove log file.

Parameters

in	<i>filePath</i>	Path to the log file.
----	-----------------	-----------------------

Return values

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

5.1.2.51 TYRemoveLogServer()

```
TY_CAPI TYRemoveLogServer (
    const char * protocol,
    const char * ip,
    uint16_t port )
```

Remove log server.

Parameters

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

Return values

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

5.1.2.52 TYSendSoftTrigger()

```
TY_CAPI TYSendSoftTrigger (
    TY_DEV_HANDLE hDevice )
```

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

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	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: 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 TYUpdateDeviceList
<i>TY_STATUS_INVALID_FEATURE</i>	Not support soft trigger.
<i>TY_STATUS_IDLE</i>	Camera device is not started Suggestions: Please start the camera device first Like this: TYStartCapture(hDevice); TYSendSoftTrigger(hDevice);
<i>TY_STATUS_WRONG_MODE</i>	Not in trigger mode.
<i>TY_STATUS_DEVICE_ERROR</i>	Failed to send soft trigger Suggestions: Possible reasons: 1.Camera device is abnormal and cannot send soft trigger. 2.Network communication is abnormal, please check whether the network is normal

Return values

<i>TY_STATUS_BUSY</i>	<p>Failed to send soft trigger</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.Camera is busy, the last soft trigger is not completed, please tr
-----------------------	---

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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
in	<i>value</i>	Bool value.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYSetBool called with invalid device handle</p> <p>Suggestions:</p> <p>Please check device handle</p> <p>Like this:</p> <pre>TYSetBool(hDevice, componentID, featureID, value);</pre> <p>^ is invalid</p> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions:</p> <p>Please check componentID parameter</p> <p>Like this:</p> <pre>TYSetBool(hDevice, componentID, featureID, value);</pre> <p>^ is invalid</p> <p>componentID should be the value returned by TYGetComponentIDs</p> <p>You can also view the components of the camera by obtaining the xn</p>

Return values

<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID Suggestions: Please check featureID parameter Like this: <code>TYSetBool(hDevice, componentID, featureID, value);</code> ^ is invalid You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetFeatures You can also view the features of the camera device by obtaining the feature list
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	Feature type mismatch Suggestions: Please check the feature type Like this: <code>TYSetBool(hDevice, componentID, featureID, value);</code> ^ type mismatch The feature type you entered does not match. You can use TYFeatureType to get the feature type
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.
<i>TY_STATUS_TIMEOUT</i>	Failed to set bool feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network is connected
<i>TY_STATUS_DEVICE_ERROR</i>	Failed to set bool feature Suggestions: Possible reasons: 1.The feature of the camera device is not available or not implemented 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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
out	<i>pBuffer</i>	Byte buffer.
in	<i>bufferSize</i>	Size of buffer.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYSetByteArray called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <code>TYSetByteArray(hDevice, componentID, featureID, pBuffer, bufferSize)</code> ^ 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 TYUpdateDeviceList</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <code>TYSetByteArray(hDevice, componentID, featureID, pBuffer, bufferSize)</code> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the xrtDeviceList</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <code>TYSetByteArray(hDevice, componentID, featureID, pBuffer, bufferSize)</code> ^ is invalid You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetFeatureIDs You can also view the features of the camera device by obtaining the xrtDeviceList</p>
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <code>TYSetByteArray(hDevice, componentID, featureID, pBuffer, bufferSize)</code> ^ type mismatch The feature type you entered does not match. You can use TYFeatureType to get the feature type</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYSetByteArray called with NULL pointer</p> <p>Suggestions: Please check your code Like this: <code>TYSetByteArray(hDevice, componentID, featureID, pBuffer, bufferSize)</code> ^ is NULL</p>
<i>TY_STATUS_WRONG_SIZE</i>	<p>Array size mismatch</p> <p>Suggestions: Please check the array size Like this: <code>TYSetByteArray(hDevice, componentID, featureID, pBuffer, bufferSize)</code> ^ is incorrect The array size you entered does not match</p>
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to set byte array feature</p> <p>Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network is connected</p>

Return values

<i>TY_STATUS_DEVICE_ERROR</i>	Failed to set byte array feature Suggestions: Possible reasons: 1.The feature of the camera device is not available or not implemented 2.Camera device is abnormal and cannot set byte array feature
-------------------------------	--

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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
in	<i>value</i>	Enum value.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	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 correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated device list by calling TYUpdateDeviceList
<i>TY_STATUS_INVALID_COMPONENT</i>	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

<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID Suggestions: Please check featureID parameter Like this: <pre>TYSetEnum(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 TYGetEnumEntryInfo You can also view the features of the camera device by obtaining the feature list
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	Feature type mismatch Suggestions: Please check the feature type Like this: <pre>TYSetEnum(hDevice, componentID, featureID, value);</pre> ^ type mismatch The feature type you entered does not match. You can use TYFeatureType to get the feature type
<i>TY_STATUS_INVALID_PARAMETER</i>	Out of range Suggestions: Please check the value Like this: <pre>TYSetEnum(hDevice, componentID, featureID, value);</pre> ^ is out of range The value is out of range, please use TYGetEnumEntryInfo to get the feature value range
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.
<i>TY_STATUS_TIMEOUT</i>	Failed to set enum feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network is connected
<i>TY_STATUS_DEVICE_ERROR</i>	Failed to set enum feature Suggestions: Possible reasons: 1.The feature of the camera device is not available or not implemented 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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
in	<i>value</i>	Float value.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYSetFloat called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <code>TYSetFloat(hDevice, componentID, featureID, value);</code> ^ 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</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <code>TYSetFloat(hDevice, componentID, featureID, value);</code> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the xn</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <code>TYSetFloat(hDevice, componentID, featureID, value);</code> ^ is invalid You entered an invalid featureID parameter You can get a list of features of the camera device through TYGetf You can also view the features of the camera device by obtaining t</p>
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <code>TYSetFloat(hDevice, componentID, featureID, value);</code> ^ type mismatch The feature type you entered does not match. You can use TYFeatur</p>
<i>TY_STATUS_OUT_OF_RANGE</i>	<p>Out of range</p> <p>Suggestions: Please check the value Like this: <code>TYSetFloat(hDevice, componentID, featureID, value);</code> ^ is out of range The value is out of range, please use TYGetFloatRange to get the r</p>
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to set float feature</p> <p>Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne</p>
<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to set float feature</p> <p>Suggestions: Possible reasons: 1.The feature of the camera device is not available or not imple 2.Camera device is abnormal and cannot set float feature</p>

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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
in	<i>value</i>	Integer value.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYSetInt called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <pre>TYSetInt(hDevice, componentID, featureID, value);</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 calling TYUpd</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <pre>TYSetInt(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 x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <pre>TYSetInt(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 TYGetf You can also view the features of the camera device by obtaining t</p>
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.

Return values

<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <code>TYSetInt(hDevice, componentID, featureID, value);</code> ^ type mismatch</p> <p>The feature type you entered does not match. You can use TYFeature</p>
<i>TY_STATUS_OUT_OF_RANGE</i>	<p>Out of range</p> <p>Suggestions: Please check the value Like this: <code>TYSetInt(hDevice, componentID, featureID, value);</code> ^ is out of range</p> <p>The value is out of range, please use TYGetIntRange to get the ran</p>
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to set int feature</p> <p>Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne</p>
<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to set int feature</p> <p>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</p>

5.1.2.58 TYSetLogLevel()

```
TY_CAPI TYSetLogLevel (
    TY LOG LEVEL lvl )
```

Set log level.

Parameters

in	$ v $	Log level.
----	-------	------------

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

5.1.2.59 TYSetLogPrefix()

```
TY_CAPI TYSetLogPrefix (
    const char * prefix )
```

set log prefix

Parameters

in	<i>prefix</i>	Prefix string.
----	---------------	----------------

Return values

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

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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
in	<i>buffer</i>	String buffer.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	TYSetString called with invalid device handle Suggestions: Please check device handle Like this: <pre>TYSetString(hDevice, componentID, featureID, pBuffer);</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 calling TYUp

Return values

<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions:</p> <p>Please check componentID parameter</p> <p>Like this:</p> <pre>TYSetString(hDevice, componentID, featureID, pBuffer);</pre> <p>^ is invalid</p> <p>componentID should be the value returned by TYGetComponentIDs</p> <p>You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions:</p> <p>Please check featureID parameter</p> <p>Like this:</p> <pre>TYSetString(hDevice, componentID, featureID, pBuffer);</pre> <p>^ is invalid</p> <p>You entered an invalid featureID parameter</p> <p>You can get a list of features of the camera device through TYGetf</p> <p>You can also view the features of the camera device by obtaining t</p>
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions:</p> <p>Please check the feature type</p> <p>Like this:</p> <pre>TYSetString(hDevice, componentID, featureID, pBuffer);</pre> <p>^ type mismatch</p> <p>The feature type you entered does not match. You can use TYFeatur</p>
<i>TY_STATUS_NULL_POINTER</i>	<p>TYSetString called with NULL pointer</p> <p>Suggestions:</p> <p>Please check your code</p> <p>Like this:</p> <pre>TYSetString(hDevice, componentID, featureID, pBuffer);</pre> <p>^ is NULL</p>
<i>TY_STATUS_OUT_OF_RANGE</i>	Input string is too long.
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.
<i>TY_STATUS_TIMEOUT</i>	<p>Failed to set string feature</p> <p>Suggestions:</p> <p>Possible reasons:</p> <p>1.Network communication is abnormal, please check whether the n</p>
<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to set string feature</p> <p>Suggestions:</p> <p>Possible reasons:</p> <p>1.The feature of the camera device is not available or not impl</p> <p>2.Camera device is abnormal and cannot set string feature</p>

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 )
```

Set value of struct.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYSetStruct called with invalid device handle</p> <p>Suggestions: Please check device handle Like this: <pre>TYSetStruct(hDevice, componentID, featureID, pStruct, structSize)</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 calling TYUpd</p>
<i>TY_STATUS_INVALID_COMPONENT</i>	<p>Invalid component ID</p> <p>Suggestions: Please check componentID parameter Like this: <pre>TYSetStruct(hDevice, componentID, featureID, pStruct, structSize)</pre> ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the x</p>
<i>TY_STATUS_INVALID_FEATURE</i>	<p>Invalid feature ID</p> <p>Suggestions: Please check featureID parameter Like this: <pre>TYSetStruct(hDevice, componentID, featureID, pStruct, structSize)</pre> ^ 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 t</p>
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	<p>Feature type mismatch</p> <p>Suggestions: Please check the feature type Like this: <pre>TYSetStruct(hDevice, componentID, featureID, pStruct, structSize)</pre> ^ type mismatch The feature type you entered does not match. You can use TYFeature</p>

Return values

<i>TY_STATUS_NULL_POINTER</i>	TYSetStruct called with NULL pointer Suggestions: Please check your code Like this: <pre>TYSetStruct(hDevice, componentID, featureID, pStruct, structSize, ^ is NULL</pre>
<i>TY_STATUS_WRONG_SIZE</i>	Struct size mismatch Suggestions: Please check the struct size Like this: <pre>TYSetStruct(hDevice, componentID, featureID, pStruct, structSize, ^ is invalid</pre> The struct size you entered does not match
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.
<i>TY_STATUS_TIMEOUT</i>	Failed to set struct feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network is normal
<i>TY_STATUS_DEVICE_ERROR</i>	Failed to set struct feature Suggestions: Possible reasons: 1.The feature of the camera device is not available or not implemented 2.Camera device is abnormal and cannot set struct feature

5.1.2.62 TYStartCapture()

```
TY_CAPI TYStartCapture (
    TY_DEV_HANDLE hDevice )
```

Start capture.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	TYStartCapture called with invalid device handle Suggestions: Please check device handle Like this: <pre>TYStartCapture(hDevice);</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 calling TYUpdateDeviceList

Return values

<i>TY_STATUS_INVALID_COMPONENT</i>	<p>No components are enabled</p> <p>Suggestions:</p> <p>Please enable the components of the camera device first</p> <p>Like this:</p> <pre>TYEnableComponents(hDevice, componentIDs); TYStartCapture(hDevice);</pre>
<i>TY_STATUS_BUSY</i>	<p>Camera device has been started.</p> <p>Suggestions:</p> <p>Please stop the camera device first</p> <p>Like this:</p> <pre>TYStopCapture(hDevice); TYStartCapture(hDevice);</pre>
<i>TY_STATUS_DEVICE_ERROR</i>	<p>Failed to start camera</p> <p>Suggestions:</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 1.Camera device is abnormal and cannot start the camera. 2.Network communication is abnormal, please check whether the network is connected. 3.Camera is busy, please try again

5.1.2.63 TYStopCapture()

```
TY_CAPI TYStopCapture (
    TY_DEV_HANDLE hDevice )
```

Stop capture.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	<p>TYStopCapture called with invalid device handle</p> <p>Suggestions:</p> <p>Please check device handle</p> <p>Like this:</p> <pre>TYStopCapture(hDevice); ^ is invalid</pre> <p>The hDevice parameter you input is not recorded</p> <p>Possible reasons:</p> <ol style="list-style-type: none"> 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 TYUpdatedDeviceList

Return values

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

5.1.2.64 TYUpdateAllDeviceList()

```
TY_CAPI TYUpdateAllDeviceList (
    void )
```

Update current connected devices.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.

5.1.2.65 TYUpdateDeviceList()

```
TY_CAPI TYUpdateDeviceList (
    TY\_INTERFACE\_HANDLE ifaceHandle )
```

Update current connected devices.

Parameters

in	<i>ifaceHandle</i>	Interface handle.
----	--------------------	-------------------

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.

Return values

<i>TY_STATUS_INVALID_INTERFACE</i>	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 correct handle 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling TYU
------------------------------------	---

5.1.2.66 TYUpdateInterfaceList()

```
TY_CAPI TYUpdateInterfaceList (
    void )
```

Update current interfaces. call before TYGetInterfaceList.

Return values

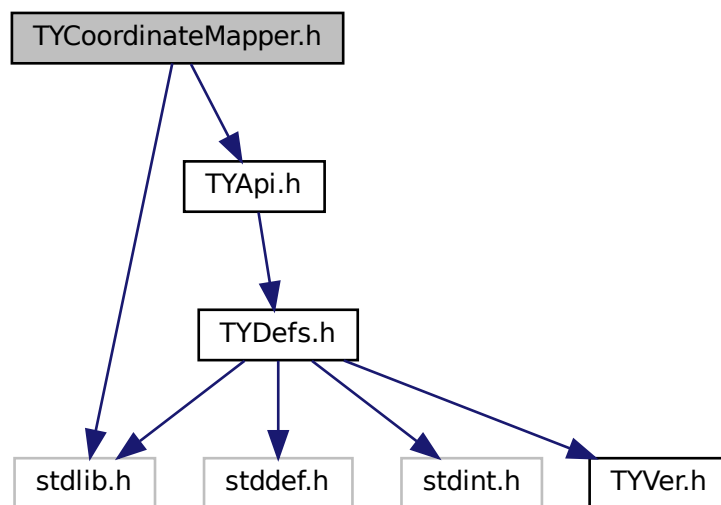
<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.

5.2 TYCoordinateMapper.h File Reference

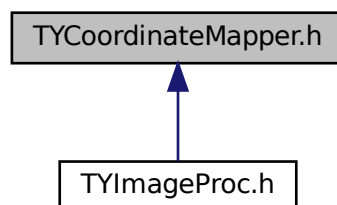
Coordinate Conversion API.

```
#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.

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

```
#define TYMAP_CHECKRET (
    f,
    bufToFree )
```

Value:

```
do{ \
    TY_STATUS err = (f); \
    if(err){ \
        if(bufToFree) \
            free(bufToFree); \
        return err; \
    } \
} while (0)
```

Definition at line 274 of file TYCoordinateMapper.h.

5.2.3 Function Documentation

5.2.3.1 TYDepthImageFillEmptyRegion()

```
TY_CAPI TYDepthImageFillEmptyRegion (
    uint16_t * depth,
    uint32_t depthW,
    uint32_t depthH )
```

Fill depth image empty region.

Parameters

in	<i>depth</i>	Depth image pixels.
in	<i>depthW</i>	Width of current depth image.
in	<i>depthH</i>	Height of current depth image.

5.2.3.2 TYInvertExtrinsic()

```
TY_CAPI TYInvertExtrinsic (
    const TY_CAMERA_EXTRINSIC * orgExtrinsic,
    TY_CAMERA_EXTRINSIC * invExtrinsic )
```

Calculate 4x4 extrinsic matrix's inverse matrix.

Parameters

in	<i>orgExtrinsic</i>	Input extrinsic matrix.
out	<i>invExtrinsic</i>	Inverse matrix.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_ERROR</i>	Calculation failed.

5.2.3.3 TYMapDepthImageToPoint3d()

```

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).

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

5.2.3.4 TYMapDepthToPoint3d()

```

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.

Parameters

in	<i>src_calib</i>	Depth image's calibration data.
in	<i>depthW</i>	Width of depth image.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

5.2.3.5 TYMapPoint3dToDepth()

```

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.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

5.2.3.6 TYMapPoint3dToDepthImage()

```

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.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

5.2.3.7 TYMapPoint3dToPoint3d()

```
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.

Parameters

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

Return values

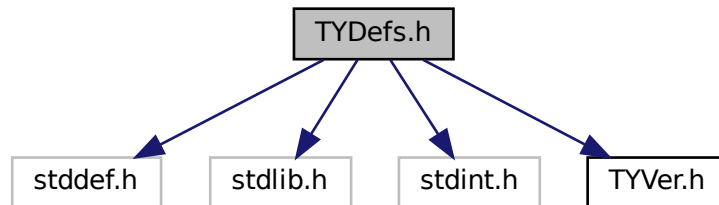
<i>TY_STATUS_OK</i>	Succeed.
---------------------	----------

5.3 TYDefs.h File Reference

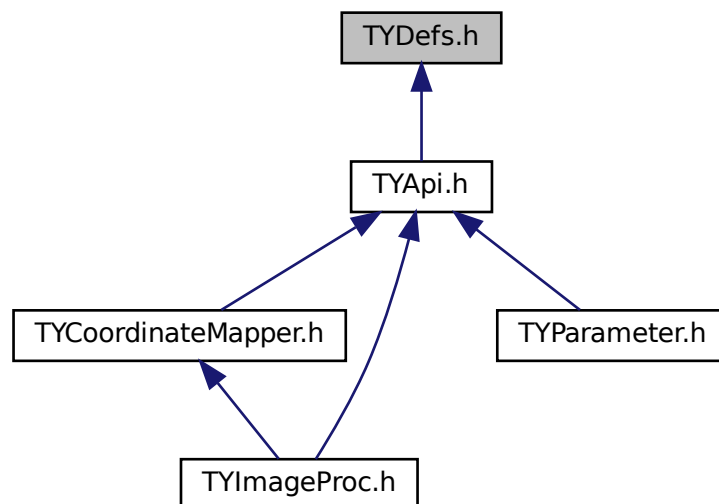
[TYDefs.h](#) includes camera control and data receiving data definitions which supports configuration for image resolution, frame rate, exposure time, gain, working mode, etc.

```
#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_PARAM TY_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_MODE0(pix, res) TY_IMAGE_MODE_##pix##_##res = TY_PIXEL_FOR↵
MAT_##pix | TY_RESOLUTION_MODE_##res`
- `#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 `TYLensOpticalType` `TYLensOpticalType`
- typedef enum `TYPixFmtList` `TYPixFmtList`
pixel format definitions
- typedef uint32_t `TYPixFmt`

- 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](#)
- 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_ERROR_XML](#) = -1025, [TY_STATUS_DEV_E](#)
[PERM](#) = -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](#) = 0x00000001, [TY_FW_ERRORCODE_CAM1_NOT_](#)
[DETECTED](#) = 0x00000002, [TY_FW_ERRORCODE_CAM2_NOT_DETECTED](#) = 0x00000004, [TY_FW_E](#)
[RRORCODE_POE_NOT_INIT](#) = 0x00000008,
[TY_FW_ERRORCODE_RECMAP_NOT_CORRECT](#) = 0x00000010, [TY_FW_ERRORCODE_LOOKUPT](#)
[ABLE_NOT_CORRECT](#) = 0x00000020, [TY_FW_ERRORCODE_DRV8899_NOT_INIT](#) = 0x00000040, [T](#)
[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_L](#)
[= 0x00040000, \[TY_COMPONENT_IR_CAM_RIGHT\]\(#\) = 0x00080000,](#)
[TY_COMPONENT_RGB_CAM_LEFT](#) = 0x00100000, [TY_COMPONENT_RGB_CAM_RIGHT](#) = 0x00200000,
[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↵**
EATURE_BOOL = 0x4000,
TY_FEATURE_STRING = 0x5000, **TY_FEATURE_BYTEARRAY** = 0x6000, **TY_FEATURE_STRUCT** =
0x7000 }

Feature Format Type definitions.

- 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_DIO_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_↵
ASER_LUT_IDX = 0x050a | TY_FEATURE_INT,
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_TARGET_↵
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_↵
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, TY_ENUM_LENS_OPTICAL_TYPE = 0x0909 | TY_FEATURE_ENUM }

```

feature for component definitions

- enum **TY_CONFIG_MODE_LIST** : uint32_t {
TY_CONFIG_MODE_PRESET0 = 0, **TY_CONFIG_MODE_PRESET1**, **TY_CONFIG_MODE_PRESET2**,
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_QUALITY_MEDIUM** = 2, **TY_DEPTH_QUALITY_HIGH** = 4 }
- enum **TY_TRIGGER_POL_LIST** : uint32_t { **TY_TRIGGER_POL_FALLINGEDGE** = 0, **TY_TRIGGER_POL_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_INTERFACE_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, **TY_STREAM_ASYNC_DEPTH_RGB** = 3,
TY_STREAM_ASYNC_ALL = 0xff }

stream async mode

- enum **TYLensOpticalType** : uint32_t { **STANDARD_LENS** = 0, **FISHEYE_LENS** = 1 }
- enum **TYPixFmtList** : uint32_t {
TYPixelFormatMono8 = 0x01080001, **TYPixelFormatBayerGBRG8** = 0x0108000A, **TYPixelFormatBayerBGGR8** = 0x0108000B, **TYPixelFormatBayerGRBG8** = 0x01080008,
TYPixelFormatBayerRGG8 = 0x01080009, **TYPixelFormatPacketMono10** = 0x010A0046, **TYPixelFormatPacketBayerGBRG10** = 0x010A0054, **TYPixelFormatPacketBayerBGGR10** = 0x010A0052,
TYPixelFormatPacketBayerGRBG10 = 0x010A0056, **TYPixelFormatPacketBayerRGG10** = 0x010A0058, **TYPixelFormatPacketMono12** = 0x010C0047, **TYPixelFormatPacketBayerGBRG12** = 0x010C0055,
TYPixelFormatPacketBayerBGGR12 = 0x010C0053, **TYPixelFormatPacketBayerGRBG12** = 0x010C0057, **TYPixelFormatPacketBayerRGG12** = 0x010C0059, **TYPixelFormatMono10** = 0x810A0046,
TYPixelFormatBayerGBRG10 = 0x810A0054, **TYPixelFormatBayerBGGR10** = 0x810A0052, **TYPixelFormatBayerGRBG10** = 0x810A0056, **TYPixelFormatBayerRGG10** = 0x810A0058,
TYPixelFormatMono12 = 0x810C0047, **TYPixelFormatBayerGBRG12** = 0x810C0055, **TYPixelFormatBayerBGGR12** = 0x810C0053, **TYPixelFormatBayerGRBG12** = 0x810C0057,
TYPixelFormatBayerRGG12 = 0x810C0059, **TYPixelFormatMono14** = 0x810E0104, **TYPixelFormatBayerGBRG14** = 0x810E0107, **TYPixelFormatBayerBGGR14** = 0x810E0108,
TYPixelFormatBayerGRBG14 = 0x810E0105, **TYPixelFormatBayerRGG14** = 0x810E0106, **TYPixelFormatMono16** = 0x01100007, **TYPixelFormatBayerGBRG16** = 0x01100030,
TYPixelFormatBayerBGGR16 = 0x01100031, **TYPixelFormatBayerGRBG16** = 0x0110002E, **TYPixelFormatBayerRGG16** = 0x0110002F, **TYPixelFormatRGB8** = 0x02180014,
TYPixelFormatBGR8 = 0x02180015, **TYPixelFormatYUV422_8** = 0x02100032, **TYPixelFormatYUV422_8_UYVY** = 0x0210001F, **TYPixelFormatYCbCr420_8_YY_CbCr_Planar** = 0x020C0113,
TYPixelFormatYCbCr420_8_YY_CrCb_Planar = 0x020C0115, **TYPixelFormatYCbCr420_8_YY_CbCr_Semiplanar** = 0x020C0112, **TYPixelFormatYCbCr420_8_YY_CrCb_Semiplanar** = 0x020C0114, **TYPixelFormatCoord3D_C16** = 0x011000B8,
TYPixelFormatCoord3D_ABC16 = 0x023000B9, **TYPixelFormatCoord3D_ABC32f** = 0x026000C0, **TYPixelFormatJPEG** = 0x82180015, **TYPixelFormatToFIRFourGroupMono16** = 0x81400016,
TYPixelFormatInvalid = 0xFFFFFFFF }

pixel format definitions

- enum **TY_PIXEL_BITS_LIST** : uint32_t {
TY_PIXEL_8BIT = 0x1 << 28, **TY_PIXEL_16BIT** = 0x2 << 28, **TY_PIXEL_24BIT** = 0x3 << 28, **TY_PIXEL_32BIT** = 0x4 << 28,
TY_PIXEL_10BIT = 0x5 << 28, **TY_PIXEL_12BIT** = 0x6 << 28, **TY_PIXEL_14BIT** = 0x7 << 28, **TY_PIXEL_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_BAYER8GB, **TY_PIXEL_FORMAT_BAYER8RGG** = TY_PIXEL_FORMAT_BAYER8BG,
 - TY_PIXEL_FORMAT_BAYER8GBRG** = TY_PIXEL_FORMAT_BAYER8GR, **TY_PIXEL_FORMAT_BAYER8BGR** = TY_PIXEL_FORMAT_BAYER8RG, **TY_PIXEL_FORMAT_CSI_MONO10** = (TY_PIXEL_10BIT | (0x0 << 24)), **TY_PIXEL_FORMAT_CSI_BAYER10GRBG** = (TY_PIXEL_10BIT | (0x1 << 24)),
 - TY_PIXEL_FORMAT_CSI_BAYER10RGG** = (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_BAYER12RGG** = (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 | (0x1 << 24)), **TY_PIXEL_FORMAT_YUYV** = (TY_PIXEL_16BIT | (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_MJPEG** = (TY_PIXEL_24BIT | (0x3 << 24)), **TY_PIXEL_FORMAT_RGB48** = (TY_PIXEL_48BIT | (0x0 << 24)), **TY_PIXEL_FORMAT_BGR48** = (TY_PIXEL_48BIT | (0x1 << 24)),
 - TY_PIXEL_FORMAT_XYZ48** = (TY_PIXEL_48BIT | (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_640x400** = (640<<12)+400, **TY_RESOLUTION_MODE_640x480** = (640<<12)+480,
 - TY_RESOLUTION_MODE_960x1280** = (960<<12)+1280, **TY_RESOLUTION_MODE_1280x720** = (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), **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, **INVLISIBLE** = 3 }
- 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_MODE0(pix, 640x360), \  
TY_DECLARE_IMAGE_MODE0(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_MODE0(pix, 1920x1440), \  
TY_DECLARE_IMAGE_MODE0(pix, 2048x1536), \  
TY_DECLARE_IMAGE_MODE0(pix, 240x96)
```

Definition at line 620 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	-GAMAy _z	GAMAz _y
GAMAx _z	1	-GAMAz _x
-GAMAx _y	GAMAy _x	1

5.3.3.3 TY_ACC_SCALE

```
typedef struct TY_ACC_SCALE TY_ACC_SCALE
```

a 3x3 matrix

.	.	.
SCALE _x	0	0
0	SCALE _y	0
0	0	SCALE _z

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	cx
0	fy	cy
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 191 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
```

@brief 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

5.3.3.15 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 377 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

.	.	.
BIASx	BIASy	BIASz

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](#)

5.3.3.23 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 431 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 176 of file TYDefs.h.

5.3.4.3 TY_FEATURE_ID_LIST

```
enum TY_FEATURE_ID_LIST : uint32_t
```

feature for component definitions

Enumerator

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	stream async switch, see TY_STREAM_ASYNC_MODE
TY_INT_CAPTURE_TIME_US	capture time in multi-ir
TY_ENUM_TIME_SYNC_TYPE	see TY_TIME_SYNC_TYPE
TY_BOOL_TIME_SYNC_READY	time sync done status

Enumerator

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 G channel.
TY_INT_B_GAIN	Gain of B channel.
TY_INT_ANALOG_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.

Enumerator

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.
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 210 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 418 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 528 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

Definition at line 546 of file TYDefs.h.

5.3.4.7 TY_RESOLUTION_MODE_LIST

```
enum TY_RESOLUTION_MODE_LIST : uint32_t
```

predefined resolution list

Enumerator

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
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 590 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

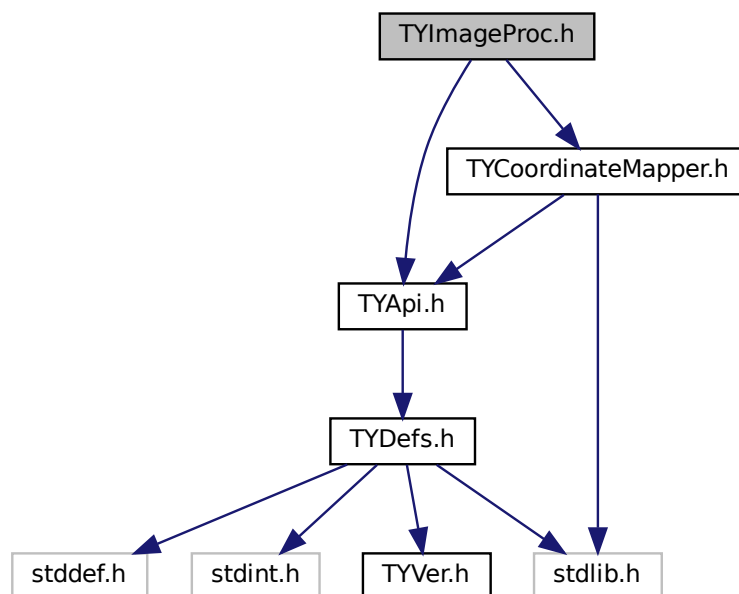
Enumerator

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 691 of file TYDefs.h.

5.4 TYImageProc.h File Reference

```
#include "TYApi.h"
#include "TYCoordinateMapper.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, const TYLensOpticalType type=STANDARD_LENS)
Do image undistortion, only support TY_PIXEL_FORMAT_MONO, TY_PIXEL_FORMAT_RGB, TY_PIXEL_FORMAT_RGBA, TY_PIXEL_FORMAT_BGR.
- 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()

```
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.

Parameters

in	<i>depthImage</i>	Pointer to depth image array.
in	<i>imageNum</i>	Depth image array size.
in, out	<i>guide</i>	Guide image.
out	<i>output</i>	Output depth image.
Generated by Doxygen	<i>param</i>	Algorithm parameters.

Return values

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

5.4.2.2 TYDepthSpeckleFilter()

```
TY_CAPI TYDepthSpeckleFilter (
    TY_IMAGE_DATA * depthImage,
    const DepthSpeckleFilterParameters * param )
```

Remove speckles on depth image.

Parameters

in, out	<i>depthImage</i>	Depth image to be processed.
in	<i>param</i>	Algorithm parameters.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NULL_POINTER</i>	Any depth, param or depth->buffer is NULL.
<i>TY_STATUS_INVALID_PARAMETER</i>	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	<i>en</i>	Enable image process acceleration switch
----	-----------	--

5.4.2.4 TYUndistortImage()

```
TY_CAPI TYUndistortImage (
    const TY_CAMERA_CALIB_INFO * srcCalibInfo,
```



```

const TY_IMAGE_DATA * srcImage,
const TY_CAMERA_INTRINSIC * cameraNewIntrinsic,
TY_IMAGE_DATA * dstImage,
const TYLensOpticalType type = STANDARD_LENS )

```

Do image undistortion, only support TY_PIXEL_FORMAT_MONO ,TY_PIXEL_FORMAT_RGB,TY_PIXEL_FORMAT_BGR.

Parameters

in	<i>srcCalibInfo</i>	Image calibration data.
in	<i>srcImage</i>	Source image.
in	<i>cameraNewIntrinsic</i>	Expected new image intrinsic, will use srcCalibInfo for new image intrinsic if set to NULL.
out	<i>dstImage</i>	Output image.

Return values

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

Index

DepthEnhenceParameters, [7](#)
DepthSpeckleFilterParameters, [7](#)

gz_header_s, [8](#)
gzFile_s, [8](#)

pattern_bin_param, [8](#)
pattern_gray_param, [9](#)
pattern_sine_param, [9](#)

TY_ACC_BIAS, [10](#)
 TYDefs.h, [121](#)
TY_ACC_MISALIGNMENT, [10](#)
 TYDefs.h, [121](#)
TY_ACC_SCALE, [11](#)
 TYDefs.h, [122](#)
TY_ACCESS_MODE_LIST
 TYDefs.h, [122](#), [128](#)
TY_AEC_ROI_PARAM, [12](#)
TY_BOOL_AUTO_AWB
 TYDefs.h, [130](#)
TY_BOOL_AUTO_EXPOSURE
 TYDefs.h, [130](#)
TY_BOOL_AUTO_GAIN
 TYDefs.h, [130](#)
TY_BOOL_BRIGHTNESS_HISTOGRAM
 TYDefs.h, [130](#)
TY_BOOL_CMOS_SYNC
 TYDefs.h, [129](#)
TY_BOOL_DEPTH_POSTPROC
 TYDefs.h, [130](#)
TY_BOOL_HDR
 TYDefs.h, [130](#)
TY_BOOL_IMU_DATA_ONOFF
 TYDefs.h, [130](#)
TY_BOOL_IR_FLASHLIGHT
 TYDefs.h, [130](#)
TY_BOOL_KEEP_ALIVE_ONOFF
 TYDefs.h, [129](#)
TY_BOOL_LASER_AUTO_CTRL
 TYDefs.h, [130](#)
TY_BOOL_RGB_FLASHLIGHT
 TYDefs.h, [130](#)
TY_BOOL_SGBM_HFILTER_HALF_WIN
 TYDefs.h, [131](#)
TY_BOOL_SGBM_LRC
 TYDefs.h, [131](#)
TY_BOOL_SGBM_MEDFILTER
 TYDefs.h, [131](#)
TY_BOOL_SGPM_EPI_EN

 TYDefs.h, [131](#)
TY_BOOL_SGPM_ORDER_FILTER_EN
 TYDefs.h, [131](#)
TY_BOOL_TIME_SYNC_READY
 TYDefs.h, [129](#)
TY_BOOL_TOF_ANTI_INTERFERENCE
 TYDefs.h, [131](#)
TY_BOOL_TRIGGER_OUT_IO
 TYDefs.h, [129](#)
TY_BOOL_UNDISTORTION
 TYDefs.h, [130](#)
TY_BYTEARRAY_ATTR, [12](#)
 TYDefs.h, [122](#)
 unit_size, [12](#)
 valid_size, [13](#)
TY_BYTEARRAY_CUSTOM_BLOCK
 TYDefs.h, [129](#)
TY_BYTEARRAY_HDR_PARAMETER
 TYDefs.h, [130](#)
TY_BYTEARRAY_ISP_BLOCK
 TYDefs.h, [129](#)
TY_CAMERA_CALIB_INFO, [13](#)
 TYDefs.h, [122](#)
TY_CAMERA_DISTORTION, [14](#)
 TYDefs.h, [123](#)
TY_CAMERA_EXTRINSIC, [14](#)
 TYDefs.h, [123](#)
TY_CAMERA_INTRINSIC, [15](#)
 TYDefs.h, [123](#)
TY_CAMERA_ROTATION, [16](#)
 TYDefs.h, [124](#)
TY_CAMERA_STATISTICS, [16](#)
TY_CAMERA_TO_IMU, [17](#)
 TYDefs.h, [124](#)
TY_COMPONENT_BRIGHT_HISTO
 TYDefs.h, [128](#)
TY_COMPONENT_DEPTH_CAM
 TYDefs.h, [128](#)
TY_COMPONENT_DEVICE
 TYDefs.h, [128](#)
TY_COMPONENT_ID
 TYDefs.h, [124](#)
TY_COMPONENT_IMU
 TYDefs.h, [128](#)
TY_COMPONENT_IR_CAM_LEFT
 TYDefs.h, [128](#)
TY_COMPONENT_IR_CAM_RIGHT
 TYDefs.h, [128](#)
TY_COMPONENT_LASER

TYDefs.h, [128](#)
TY_COMPONENT_RGB_CAM
TYDefs.h, [128](#)
TY_COMPONENT_RGB_CAM_LEFT
TYDefs.h, [128](#)
TY_COMPONENT_RGB_CAM_RIGHT
TYDefs.h, [128](#)
TY_COMPONENT_STORAGE
TYDefs.h, [128](#)
TY_DECLARE_IMAGE_MODE1
TYDefs.h, [121](#)
TY_DEVICE_BASE_INFO, [17](#)
TYDefs.h, [125](#)
TY_DEVICE_COMPONENT_LIST
TYDefs.h, [125](#), [128](#)
TY_DEVICE_NET_INFO, [18](#)
TY_DEVICE_USB_INFO, [19](#)
TY_DI_WORKMODE, [19](#)
TY_DO_WORKMODE, [20](#)
TY_ENUM_DEPTH_QUALITY
TYDefs.h, [131](#)
TY_ENUM_ENTRY, [20](#)
TYDefs.h, [125](#)
TY_ENUM_IMAGE_MODE
TYDefs.h, [129](#)
TY_ENUM_IMU_FPS
TYDefs.h, [130](#)
TY_ENUM_STREAM_ASYNC
TYDefs.h, [129](#)
TY_ENUM_TIME_SYNC_TYPE
TYDefs.h, [129](#)
TY_ENUM_TRIGGER_POL
TYDefs.h, [129](#)
TY_EVENT_INFO, [21](#)
TY_FEATURE_ID
TYDefs.h, [125](#)
TY_FEATURE_ID_LIST
TYDefs.h, [129](#)
TY_FEATURE_INFO, [21](#)
TY_FLOAT_EXPOSURE_TIME_US
TYDefs.h, [130](#)
TY_FLOAT_RANGE, [22](#)
TYDefs.h, [126](#)
TY_FLOAT_SCALE_UNIT
TYDefs.h, [129](#)
TY_FLOAT_SGPM_EPI_HS
TYDefs.h, [131](#)
TY_FRAME_DATA, [22](#)
TY_GYRO_BIAS, [23](#)
TYDefs.h, [126](#)
TY_GYRO_MISALIGNMENT, [23](#)
TYDefs.h, [126](#)
TY_GYRO_SCALE, [24](#)
TYDefs.h, [127](#)
TY_IMAGE_DATA, [25](#)
TY_IMU_DATA, [25](#)
TY_INT_ANALOG_GAIN
TYDefs.h, [130](#)
TY_INT_B_GAIN
TYDefs.h, [130](#)
TY_INT_CAPTURE_TIME_US
TYDefs.h, [129](#)
TY_INT_DEPTH_MAX_MM
TYDefs.h, [131](#)
TY_INT_DEPTH_MIN_MM
TYDefs.h, [131](#)
TY_INT_EXPOSURE_TIME
TYDefs.h, [130](#)
TY_INT_FILTER_THRESHOLD
TYDefs.h, [131](#)
TY_INT_FRAME_PER_TRIGGER
TYDefs.h, [129](#)
TY_INT_G_GAIN
TYDefs.h, [130](#)
TY_INT_GAIN
TYDefs.h, [130](#)
TY_INT_HEIGHT
TYDefs.h, [129](#)
TY_INT_IR_FLASHLIGHT_INTENSITY
TYDefs.h, [130](#)
TY_INT_KEEP_ALIVE_TIMEOUT
TYDefs.h, [129](#)
TY_INT_LASER_POWER
TYDefs.h, [130](#)
TY_INT_LINK_CMD_TIMEOUT
TYDefs.h, [129](#)
TY_INT_MAX_SPECKLE_DIFF
TYDefs.h, [131](#)
TY_INT_MAX_SPECKLE_SIZE
TYDefs.h, [131](#)
TY_INT_NTP_SERVER_IP
TYDefs.h, [129](#)
TY_INT_PACKET_DELAY
TYDefs.h, [129](#)
TY_INT_R_GAIN
TYDefs.h, [130](#)
TY_INT_RANGE, [26](#)
TY_INT_RGB_FLASHLIGHT_INTENSITY
TYDefs.h, [130](#)
TY_INT_SGBM_DISPARITY_NUM
TYDefs.h, [130](#)
TY_INT_SGBM_DISPARITY_OFFSET
TYDefs.h, [130](#)
TY_INT_SGBM_IMAGE_NUM
TYDefs.h, [130](#)
TY_INT_SGBM_LRC_DIFF
TYDefs.h, [131](#)
TY_INT_SGBM_MATCH_WIN_HEIGHT
TYDefs.h, [131](#)
TY_INT_SGBM_MATCH_WIN_WIDTH
TYDefs.h, [131](#)
TY_INT_SGBM_MEDFILTER_THRESH
TYDefs.h, [131](#)
TY_INT_SGBM_SEMI_PARAM_P1
TYDefs.h, [131](#)
TY_INT_SGBM_SEMI_PARAM_P1_SCALE

- TYDefs.h, [131](#)
- TY_INT_SGBM_SEMI_PARAM_P2
 - TYDefs.h, [131](#)
- TY_INT_SGBM_TEXTURE_OFFSET
 - TYDefs.h, [131](#)
- TY_INT_SGBM_TEXTURE_THRESH
 - TYDefs.h, [131](#)
- TY_INT_SGBM_UNIQUE_ABSDIFF
 - TYDefs.h, [131](#)
- TY_INT_SGBM_UNIQUE_FACTOR
 - TYDefs.h, [131](#)
- TY_INT_SGBM_UNIQUE_MAX_COST
 - TYDefs.h, [131](#)
- TY_INT_SGPM_EPI_CH0
 - TYDefs.h, [131](#)
- TY_INT_SGPM_EPI_CH1
 - TYDefs.h, [131](#)
- TY_INT_SGPM_EPI_HF
 - TYDefs.h, [131](#)
- TY_INT_SGPM_EPI_THRESH
 - TYDefs.h, [131](#)
- TY_INT_SGPM_NORMAL_PHASE_OFFSET
 - TYDefs.h, [131](#)
- TY_INT_SGPM_NORMAL_PHASE_SCALE
 - TYDefs.h, [131](#)
- TY_INT_SGPM_ORDER_FILTER_CHN
 - TYDefs.h, [131](#)
- TY_INT_SGPM_PHASE_NUM
 - TYDefs.h, [131](#)
- TY_INT_SGPM_REF_PHASE_OFFSET
 - TYDefs.h, [131](#)
- TY_INT_SGPM_REF_PHASE_SCALE
 - TYDefs.h, [131](#)
- TY_INT_TOF_ANTI_SUNLIGHT_INDEX
 - TYDefs.h, [131](#)
- TY_INT_TOF_CHANNEL
 - TYDefs.h, [131](#)
- TY_INT_TOF_HDR_RATIO
 - TYDefs.h, [130](#)
- TY_INT_TOF_JITTER_THRESHOLD
 - TYDefs.h, [130](#)
- TY_INT_TOF_MODULATION_THRESHOLD
 - TYDefs.h, [131](#)
- TY_INT_TRIGGER_DELAY_US
 - TYDefs.h, [129](#)
- TY_INT_TRIGGER_DURATION_US
 - TYDefs.h, [129](#)
- TY_INT_WIDTH
 - TYDefs.h, [129](#)
- TY_INTERFACE_INFO, [26](#)
 - TYDefs.h, [127](#)
- TY_INTERFACE_TYPE_LIST
 - TYDefs.h, [127](#), [131](#)
- TY_LASER_PARAM, [27](#)
- TY_LASER_PATTERN_PARAM, [27](#)
- TY_PHC_GROUP_ATTR, [28](#)
- TY_PHC_GROUP_ATTR::phc_group_attr, [10](#)
- TY_PIXEL_BITS_LIST
 - TYDefs.h, [127](#), [132](#)
- TY_PIXEL_COLOR_DESC, [29](#)
- TY_PIXEL_DESC, [29](#)
- TY_PIXEL_FORMAT_BAYER8BG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_BAYER8GB
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_BAYER8GR
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_BAYER8RG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_BGR
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_BGR48
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER10BGGR
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER10GBRG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER10GRBG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER10RGGG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER12BGGR
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER12GBRG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER12GRBG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER12RGGG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_MONO10
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_CSI_MONO12
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_DEPTH16
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_JPEG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_LIST
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_MJPG
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_MONO
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_MONO16
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_RGB
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_RGB48
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_TOF_IR_MONO16
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_XYZ48
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_YUYV
 - TYDefs.h, [132](#)
- TY_PIXEL_FORMAT_YVYU
 - TYDefs.h, [132](#)

- TYDefs.h, [132](#)
- TY_RESOLUTION_MODE_1280x720
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_1280x800
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_1280x960
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_1600x1200
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_160x100
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_160x120
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_1920x1080
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_1920x1440
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_2048x1536
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_240x320
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_240x96
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_2560x1920
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_2592x1944
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_320x180
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_320x200
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_320x240
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_480x640
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_640x360
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_640x400
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_640x480
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_800x600
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_960x1280
 - TYDefs.h, [133](#)
- TY_RESOLUTION_MODE_LIST
 - TYDefs.h, [133](#)
- TY_STRUCT_AEC_ROI
 - TYDefs.h, [130](#)
- TY_STRUCT_CAM_CALIB_DATA
 - TYDefs.h, [129](#)
- TY_STRUCT_CAM_DISTORTION
 - TYDefs.h, [129](#)
- TY_STRUCT_CAM_INTRINSIC
 - TYDefs.h, [129](#)
- TY_STRUCT_CAM_RECTIFIED_INTRI
 - TYDefs.h, [129](#)
- TY_STRUCT_CAM_RECTIFIED_ROTATION
 - TYDefs.h, [129](#)
- TY_STRUCT_CAM_STATISTICS
 - TYDefs.h, [129](#)
- TY_STRUCT_DI0_WORKMODE
 - TYDefs.h, [130](#)
- TY_STRUCT_DI1_WORKMODE
 - TYDefs.h, [130](#)
- TY_STRUCT_DI2_WORKMODE
 - TYDefs.h, [130](#)
- TY_STRUCT_DO0_WORKMODE
 - TYDefs.h, [130](#)
- TY_STRUCT_DO1_WORKMODE
 - TYDefs.h, [130](#)
- TY_STRUCT_DO2_WORKMODE
 - TYDefs.h, [130](#)
- TY_STRUCT_EXTRINSIC_TO_DEPTH
 - TYDefs.h, [129](#)
- TY_STRUCT_EXTRINSIC_TO_IR_LEFT
 - TYDefs.h, [129](#)
- TY_STRUCT_FLOOD_ENABLE_BY_IDX
 - TYDefs.h, [130](#)
- TY_STRUCT_FLOOD_POWER_BY_IDX
 - TYDefs.h, [130](#)
- TY_STRUCT_IMU_ACC_BIAS
 - TYDefs.h, [130](#)
- TY_STRUCT_IMU_ACC_MISALIGNMENT
 - TYDefs.h, [130](#)
- TY_STRUCT_IMU_ACC_SCALE
 - TYDefs.h, [130](#)
- TY_STRUCT_IMU_CAM_TO_IMU
 - TYDefs.h, [130](#)
- TY_STRUCT_IMU_GYRO_BIAS
 - TYDefs.h, [130](#)
- TY_STRUCT_IMU_GYRO_MISALIGNMENT
 - TYDefs.h, [130](#)
- TY_STRUCT_IMU_GYRO_SCALE
 - TYDefs.h, [130](#)
- TY_STRUCT_LASER_ENABLE_BY_IDX
 - TYDefs.h, [130](#)
- TY_STRUCT_LASER_POWER_BY_IDX
 - TYDefs.h, [130](#)
- TY_STRUCT_PHC_GROUP_ATTR
 - TYDefs.h, [131](#)
- TY_STRUCT_TOF_FREQ
 - TYDefs.h, [131](#)
- TY_STRUCT_TRIGGER_PARAM
 - TYDefs.h, [129](#)
- TY_STRUCT_TRIGGER_PARAM_EX
 - TYDefs.h, [129](#)
- TY_STRUCT_TRIGGER_TIMER_LIST
 - TYDefs.h, [129](#)
- TY_STRUCT_TRIGGER_TIMER_PERIOD
 - TYDefs.h, [129](#)
- TY_TEMP_DATA, [30](#)
- TY_TOF_FREQ, [30](#)
- TY_TRIGGER_MODE_LIST
 - TYDefs.h, [127](#), [133](#)
- TY_TRIGGER_MODE_M_PER

- TYDefs.h, [134](#)
- TY_TRIGGER_MODE_M_SIG
 - TYDefs.h, [134](#)
- TY_TRIGGER_MODE_OFF
 - TYDefs.h, [134](#)
- TY_TRIGGER_MODE_PER_PASS
 - TYDefs.h, [134](#)
- TY_TRIGGER_MODE_PER_PASS2
 - TYDefs.h, [134](#)
- TY_TRIGGER_MODE_SIG_PASS
 - TYDefs.h, [134](#)
- TY_TRIGGER_MODE_SLAVE
 - TYDefs.h, [134](#)
- TY_TRIGGER_PARAM, [30](#)
- TY_TRIGGER_PARAM_EX, [31](#)
- TY_TRIGGER_TIMER_LIST, [31](#)
- TY_TRIGGER_TIMER_PERIOD, [32](#)
- TY_VECT_3F, [32](#)
- TY_VERSION_INFO, [32](#)
- TYApi.h, [35](#)
 - TYAppendLogToFile, [39](#)
 - TYAppendLogToServer, [40](#)
 - TYClearBufferQueue, [41](#)
 - TYCloseDevice, [41](#)
 - TYCloseInterface, [42](#)
 - TYDeinitLib, [42](#)
 - TYDisableComponents, [43](#)
 - TYEnableComponents, [44](#)
 - TYEnqueueBuffer, [45](#)
 - TYErrorString, [46](#)
 - TYFetchFrame, [46](#)
 - TYForceDeviceIP, [47](#)
 - TYGetBool, [48](#)
 - TYGetByteArray, [50](#)
 - TYGetByteArrayAttr, [51](#)
 - TYGetByteArraySize, [53](#)
 - TYGetComponentIDs, [54](#)
 - TYGetDeviceFeatureInfo, [55](#)
 - TYGetDeviceFeatureNumber, [56](#)
 - TYGetDeviceInfo, [57](#)
 - TYGetDeviceInterface, [58](#)
 - TYGetDeviceList, [59](#)
 - TYGetDeviceNumber, [59](#)
 - TYGetDeviceXML, [60](#)
 - TYGetDeviceXMLSize, [61](#)
 - TYGetEnabledComponents, [62](#)
 - TYGetEnum, [63](#)
 - TYGetEnumEntryCount, [64](#)
 - TYGetEnumEntryInfo, [65](#)
 - TYGetFeatureInfo, [66](#)
 - TYGetFloat, [67](#)
 - TYGetFloatRange, [69](#)
 - TYGetFrameBufferSize, [70](#)
 - TYGetInt, [71](#)
 - TYGetInterfaceList, [72](#)
 - TYGetInterfaceNumber, [73](#)
 - TYGetIntRange, [74](#)
 - TYGetString, [75](#)
 - TYGetStringLength, [76](#)
 - TYGetStruct, [78](#)
 - TYHasDevice, [80](#)
 - TYHasFeature, [81](#)
 - TYHasInterface, [82](#)
 - TYLibVersion, [82](#)
 - TYOpenDevice, [83](#)
 - TYOpenDeviceWithIP, [84](#)
 - TYOpenInterface, [86](#)
 - TYRegisterEventCallback, [86](#)
 - TYRegisterImuCallback, [87](#)
 - TYRemoveLogFile, [88](#)
 - TYRemoveLogServer, [88](#)
 - TYSendSoftTrigger, [89](#)
 - TYSetBool, [90](#)
 - TYSetByteArray, [91](#)
 - TYSetEnum, [93](#)
 - TYSetFloat, [94](#)
 - TYSetInt, [96](#)
 - TYSetLogLevel, [97](#)
 - TYSetLogPrefix, [97](#)
 - TYSetString, [98](#)
 - TYSetStruct, [99](#)
 - TYStartCapture, [101](#)
 - TYStopCapture, [102](#)
 - TYUpdateAllDeviceList, [103](#)
 - TYUpdateDeviceList, [103](#)
 - TYUpdateInterfaceList, [104](#)
- TYAppendLogToFile
 - TYApi.h, [39](#)
- TYAppendLogToServer
 - TYApi.h, [40](#)
- TYClearBufferQueue
 - TYApi.h, [41](#)
- TYCloseDevice
 - TYApi.h, [41](#)
- TYCloseInterface
 - TYApi.h, [42](#)
- TYCoordinateMapper.h, [104](#)
 - TYDepthImageFillEmptyRegion, [107](#)
 - TYInvertExtrinsic, [107](#)
 - TYMAP_CHECKRET, [106](#)
 - TYMapDepthImageToPoint3d, [108](#)
 - TYMapDepthToPoint3d, [108](#)
 - TYMapPoint3dToDepth, [109](#)
 - TYMapPoint3dToDepthImage, [109](#)
 - TYMapPoint3dToPoint3d, [110](#)
- TYDefs.h, [110](#)
 - TY_ACC_BIAS, [121](#)
 - TY_ACC_MISALIGNMENT, [121](#)
 - TY_ACC_SCALE, [122](#)
 - TY_ACCESS_MODE_LIST, [122](#), [128](#)
 - TY_BOOL_AUTO_AWB, [130](#)
 - TY_BOOL_AUTO_EXPOSURE, [130](#)
 - TY_BOOL_AUTO_GAIN, [130](#)
 - TY_BOOL_BRIGHTNESS_HISTOGRAM, [130](#)
 - TY_BOOL_CMOS_SYNC, [129](#)
 - TY_BOOL_DEPTH_POSTPROC, [130](#)

TY_BOOL_HDR, 130
TY_BOOL_IMU_DATA_ONOFF, 130
TY_BOOL_IR_FLASHLIGHT, 130
TY_BOOL_KEEP_ALIVE_ONOFF, 129
TY_BOOL_LASER_AUTO_CTRL, 130
TY_BOOL_RGB_FLASHLIGHT, 130
TY_BOOL_SGBM_HFILTER_HALF_WIN, 131
TY_BOOL_SGBM_LRC, 131
TY_BOOL_SGBM_MEDFILTER, 131
TY_BOOL_SGPM_EPI_EN, 131
TY_BOOL_SGPM_ORDER_FILTER_EN, 131
TY_BOOL_TIME_SYNC_READY, 129
TY_BOOL_TOF_ANTI_INTERFERENCE, 131
TY_BOOL_TRIGGER_OUT_IO, 129
TY_BOOL_UNDISTORTION, 130
TY_BYTEARRAY_ATTR, 122
TY_BYTEARRAY_CUSTOM_BLOCK, 129
TY_BYTEARRAY_HDR_PARAMETER, 130
TY_BYTEARRAY_ISP_BLOCK, 129
TY_CAMERA_CALIB_INFO, 122
TY_CAMERA_DISTORTION, 123
TY_CAMERA_EXTRINSIC, 123
TY_CAMERA_INTRINSIC, 123
TY_CAMERA_ROTATION, 124
TY_CAMERA_TO_IMU, 124
TY_COMPONENT_BRIGHT_HISTO, 128
TY_COMPONENT_DEPTH_CAM, 128
TY_COMPONENT_DEVICE, 128
TY_COMPONENT_ID, 124
TY_COMPONENT_IMU, 128
TY_COMPONENT_IR_CAM_LEFT, 128
TY_COMPONENT_IR_CAM_RIGHT, 128
TY_COMPONENT_LASER, 128
TY_COMPONENT_RGB_CAM, 128
TY_COMPONENT_RGB_CAM_LEFT, 128
TY_COMPONENT_RGB_CAM_RIGHT, 128
TY_COMPONENT_STORAGE, 128
TY_DECLARE_IMAGE_MODE1, 121
TY_DEVICE_BASE_INFO, 125
TY_DEVICE_COMPONENT_LIST, 125, 128
TY_ENUM_DEPTH_QUALITY, 131
TY_ENUM_ENTRY, 125
TY_ENUM_IMAGE_MODE, 129
TY_ENUM_IMU_FPS, 130
TY_ENUM_STREAM_ASYNC, 129
TY_ENUM_TIME_SYNC_TYPE, 129
TY_ENUM_TRIGGER_POL, 129
TY_FEATURE_ID, 125
TY_FEATURE_ID_LIST, 129
TY_FLOAT_EXPOSURE_TIME_US, 130
TY_FLOAT_RANGE, 126
TY_FLOAT_SCALE_UNIT, 129
TY_FLOAT_SGPM_EPI_HS, 131
TY_GYRO_BIAS, 126
TY_GYRO_MISALIGNMENT, 126
TY_GYRO_SCALE, 127
TY_INT_ANALOG_GAIN, 130
TY_INT_B_GAIN, 130
TY_INT_CAPTURE_TIME_US, 129
TY_INT_DEPTH_MAX_MM, 131
TY_INT_DEPTH_MIN_MM, 131
TY_INT_EXPOSURE_TIME, 130
TY_INT_FILTER_THRESHOLD, 131
TY_INT_FRAME_PER_TRIGGER, 129
TY_INT_G_GAIN, 130
TY_INT_GAIN, 130
TY_INT_HEIGHT, 129
TY_INT_IR_FLASHLIGHT_INTENSITY, 130
TY_INT_KEEP_ALIVE_TIMEOUT, 129
TY_INT_LASER_POWER, 130
TY_INT_LINK_CMD_TIMEOUT, 129
TY_INT_MAX_SPECKLE_DIFF, 131
TY_INT_MAX_SPECKLE_SIZE, 131
TY_INT_NTP_SERVER_IP, 129
TY_INT_PACKET_DELAY, 129
TY_INT_R_GAIN, 130
TY_INT_RGB_FLASHLIGHT_INTENSITY, 130
TY_INT_SGBM_DISPARITY_NUM, 130
TY_INT_SGBM_DISPARITY_OFFSET, 130
TY_INT_SGBM_IMAGE_NUM, 130
TY_INT_SGBM_LRC_DIFF, 131
TY_INT_SGBM_MATCH_WIN_HEIGHT, 131
TY_INT_SGBM_MATCH_WIN_WIDTH, 131
TY_INT_SGBM_MEDFILTER_THRESH, 131
TY_INT_SGBM_SEMI_PARAM_P1, 131
TY_INT_SGBM_SEMI_PARAM_P1_SCALE, 131
TY_INT_SGBM_SEMI_PARAM_P2, 131
TY_INT_SGBM_TEXTURE_OFFSET, 131
TY_INT_SGBM_TEXTURE_THRESH, 131
TY_INT_SGBM_UNIQUE_ABSDIFF, 131
TY_INT_SGBM_UNIQUE_FACTOR, 131
TY_INT_SGBM_UNIQUE_MAX_COST, 131
TY_INT_SGPM_EPI_CH0, 131
TY_INT_SGPM_EPI_CH1, 131
TY_INT_SGPM_EPI_HF, 131
TY_INT_SGPM_EPI_THRESH, 131
TY_INT_SGPM_NORMAL_PHASE_OFFSET, 131
TY_INT_SGPM_NORMAL_PHASE_SCALE, 131
TY_INT_SGPM_ORDER_FILTER_CHN, 131
TY_INT_SGPM_PHASE_NUM, 131
TY_INT_SGPM_REF_PHASE_OFFSET, 131
TY_INT_SGPM_REF_PHASE_SCALE, 131
TY_INT_TOF_ANTI_SUNLIGHT_INDEX, 131
TY_INT_TOF_CHANNEL, 131
TY_INT_TOF_HDR_RATIO, 130
TY_INT_TOF_JITTER_THRESHOLD, 130
TY_INT_TOF_MODULATION_THRESHOLD, 131
TY_INT_TRIGGER_DELAY_US, 129
TY_INT_TRIGGER_DURATION_US, 129
TY_INT_WIDTH, 129
TY_INTERFACE_INFO, 127
TY_INTERFACE_TYPE_LIST, 127, 131
TY_PIXEL_BITS_LIST, 127, 132
TY_PIXEL_FORMAT_BAYER8BG, 132
TY_PIXEL_FORMAT_BAYER8GB, 132
TY_PIXEL_FORMAT_BAYER8GR, 132

- TY_PIXEL_FORMAT_BAYER8RG, [132](#)
- TY_PIXEL_FORMAT_BGR, [132](#)
- TY_PIXEL_FORMAT_BGR48, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER10BGGR, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER10GBRG, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER10GRBG, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER10RGG, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER12BGGR, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER12GBRG, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER12GRBG, [132](#)
- TY_PIXEL_FORMAT_CSI_BAYER12RGG, [132](#)
- TY_PIXEL_FORMAT_CSI_MONO10, [132](#)
- TY_PIXEL_FORMAT_CSI_MONO12, [132](#)
- TY_PIXEL_FORMAT_DEPTH16, [132](#)
- TY_PIXEL_FORMAT_JPEG, [132](#)
- TY_PIXEL_FORMAT_LIST, [132](#)
- TY_PIXEL_FORMAT_MJPEG, [132](#)
- TY_PIXEL_FORMAT_MONO, [132](#)
- TY_PIXEL_FORMAT_MONO16, [132](#)
- TY_PIXEL_FORMAT_RGB, [132](#)
- TY_PIXEL_FORMAT_RGB48, [132](#)
- TY_PIXEL_FORMAT_TOF_IR_MONO16, [132](#)
- TY_PIXEL_FORMAT_XYZ48, [132](#)
- TY_PIXEL_FORMAT_YUYV, [132](#)
- TY_PIXEL_FORMAT_YVYU, [132](#)
- TY_RESOLUTION_MODE_1280x720, [133](#)
- TY_RESOLUTION_MODE_1280x800, [133](#)
- TY_RESOLUTION_MODE_1280x960, [133](#)
- TY_RESOLUTION_MODE_1600x1200, [133](#)
- TY_RESOLUTION_MODE_160x100, [133](#)
- TY_RESOLUTION_MODE_160x120, [133](#)
- TY_RESOLUTION_MODE_1920x1080, [133](#)
- TY_RESOLUTION_MODE_1920x1440, [133](#)
- TY_RESOLUTION_MODE_2048x1536, [133](#)
- TY_RESOLUTION_MODE_240x320, [133](#)
- TY_RESOLUTION_MODE_240x96, [133](#)
- TY_RESOLUTION_MODE_2560x1920, [133](#)
- TY_RESOLUTION_MODE_2592x1944, [133](#)
- TY_RESOLUTION_MODE_320x180, [133](#)
- TY_RESOLUTION_MODE_320x200, [133](#)
- TY_RESOLUTION_MODE_320x240, [133](#)
- TY_RESOLUTION_MODE_480x640, [133](#)
- TY_RESOLUTION_MODE_640x360, [133](#)
- TY_RESOLUTION_MODE_640x400, [133](#)
- TY_RESOLUTION_MODE_640x480, [133](#)
- TY_RESOLUTION_MODE_800x600, [133](#)
- TY_RESOLUTION_MODE_960x1280, [133](#)
- TY_RESOLUTION_MODE_LIST, [133](#)
- TY_STRUCT_AEC_ROI, [130](#)
- TY_STRUCT_CAM_CALIB_DATA, [129](#)
- TY_STRUCT_CAM_DISTORTION, [129](#)
- TY_STRUCT_CAM_INTRINSIC, [129](#)
- TY_STRUCT_CAM_RECTIFIED_INTRI, [129](#)
- TY_STRUCT_CAM_RECTIFIED_ROTATION, [129](#)
- TY_STRUCT_CAM_STATISTICS, [129](#)
- TY_STRUCT_DI0_WORKMODE, [130](#)
- TY_STRUCT_DI1_WORKMODE, [130](#)
- TY_STRUCT_DI2_WORKMODE, [130](#)
- TY_STRUCT_DO0_WORKMODE, [130](#)
- TY_STRUCT_DO1_WORKMODE, [130](#)
- TY_STRUCT_DO2_WORKMODE, [130](#)
- TY_STRUCT_EXTRINSIC_TO_DEPTH, [129](#)
- TY_STRUCT_EXTRINSIC_TO_IR_LEFT, [129](#)
- TY_STRUCT_FLOOD_ENABLE_BY_IDX, [130](#)
- TY_STRUCT_FLOOD_POWER_BY_IDX, [130](#)
- TY_STRUCT_IMU_ACC_BIAS, [130](#)
- TY_STRUCT_IMU_ACC_MISALIGNMENT, [130](#)
- TY_STRUCT_IMU_ACC_SCALE, [130](#)
- TY_STRUCT_IMU_CAM_TO_IMU, [130](#)
- TY_STRUCT_IMU_GYRO_BIAS, [130](#)
- TY_STRUCT_IMU_GYRO_MISALIGNMENT, [130](#)
- TY_STRUCT_IMU_GYRO_SCALE, [130](#)
- TY_STRUCT_LASER_ENABLE_BY_IDX, [130](#)
- TY_STRUCT_LASER_POWER_BY_IDX, [130](#)
- TY_STRUCT_PHC_GROUP_ATTR, [131](#)
- TY_STRUCT_TOF_FREQ, [131](#)
- TY_STRUCT_TRIGGER_PARAM, [129](#)
- TY_STRUCT_TRIGGER_PARAM_EX, [129](#)
- TY_STRUCT_TRIGGER_TIMER_LIST, [129](#)
- TY_STRUCT_TRIGGER_TIMER_PERIOD, [129](#)
- TY_TRIGGER_MODE_LIST, [127](#), [133](#)
- TY_TRIGGER_MODE_M_PER, [134](#)
- TY_TRIGGER_MODE_M_SIG, [134](#)
- TY_TRIGGER_MODE_OFF, [134](#)
- TY_TRIGGER_MODE_PER_PASS, [134](#)
- TY_TRIGGER_MODE_PER_PASS2, [134](#)
- TY_TRIGGER_MODE_SIG_PASS, [134](#)
- TY_TRIGGER_MODE_SLAVE, [134](#)
- TYDeinitLib
 - TYApi.h, [42](#)
- TYDepthEnhanceFilter
 - TYImageProc.h, [135](#)
- TYDepthImageFillEmptyRegion
 - TYCoordinateMapper.h, [107](#)
- TYDepthSpeckleFilter
 - TYImageProc.h, [136](#)
- TYDisableComponents
 - TYApi.h, [43](#)
- TYEnableComponents
 - TYApi.h, [44](#)
- TYEnqueueBuffer
 - TYApi.h, [45](#)
- TYEnumEntry, [33](#)
- TYErrorString
 - TYApi.h, [46](#)
- TYFetchFrame
 - TYApi.h, [46](#)
- TYForceDeviceIP
 - TYApi.h, [47](#)
- TYGetBool
 - TYApi.h, [48](#)
- TYGetByteArray
 - TYApi.h, [50](#)
- TYGetByteArrayAttr
 - TYApi.h, [51](#)
- TYGetByteArraySize

- TYApi.h, [53](#)
- TYGetComponentIDs
 - TYApi.h, [54](#)
- TYGetDeviceFeatureInfo
 - TYApi.h, [55](#)
- TYGetDeviceFeatureNumber
 - TYApi.h, [56](#)
- TYGetDeviceInfo
 - TYApi.h, [57](#)
- TYGetDeviceInterface
 - TYApi.h, [58](#)
- TYGetDeviceList
 - TYApi.h, [59](#)
- TYGetDeviceNumber
 - TYApi.h, [59](#)
- TYGetDeviceXML
 - TYApi.h, [60](#)
- TYGetDeviceXMLSize
 - TYApi.h, [61](#)
- TYGetEnabledComponents
 - TYApi.h, [62](#)
- TYGetEnum
 - TYApi.h, [63](#)
- TYGetEnumEntryCount
 - TYApi.h, [64](#)
- TYGetEnumEntryInfo
 - TYApi.h, [65](#)
- TYGetFeatureInfo
 - TYApi.h, [66](#)
- TYGetFloat
 - TYApi.h, [67](#)
- TYGetFloatRange
 - TYApi.h, [69](#)
- TYGetFrameBufferSize
 - TYApi.h, [70](#)
- TYGetInt
 - TYApi.h, [71](#)
- TYGetInterfaceList
 - TYApi.h, [72](#)
- TYGetInterfaceNumber
 - TYApi.h, [73](#)
- TYGetIntRange
 - TYApi.h, [74](#)
- TYGetString
 - TYApi.h, [75](#)
- TYGetStringLength
 - TYApi.h, [76](#)
- TYGetStruct
 - TYApi.h, [78](#)
- TYHasDevice
 - TYApi.h, [80](#)
- TYHasFeature
 - TYApi.h, [81](#)
- TYHasInterface
 - TYApi.h, [82](#)
- TYImageProc.h, [134](#)
 - TYDepthEnhanceFilter, [135](#)
 - TYDepthSpeckleFilter, [136](#)
 - TYImageProcesAcceEnable, [136](#)
 - TYUndistortImage, [136](#)
- TYImageProcesAcceEnable
 - TYImageProc.h, [136](#)
- TYInvertExtrinsic
 - TYCoordinateMapper.h, [107](#)
- TYLibVersion
 - TYApi.h, [82](#)
- TYMAP_CHECKRET
 - TYCoordinateMapper.h, [106](#)
- TYMapDepthImageToPoint3d
 - TYCoordinateMapper.h, [108](#)
- TYMapDepthToPoint3d
 - TYCoordinateMapper.h, [108](#)
- TYMapPoint3dToDepth
 - TYCoordinateMapper.h, [109](#)
- TYMapPoint3dToDepthImage
 - TYCoordinateMapper.h, [109](#)
- TYMapPoint3dToPoint3d
 - TYCoordinateMapper.h, [110](#)
- TYOpenDevice
 - TYApi.h, [83](#)
- TYOpenDeviceWithIP
 - TYApi.h, [84](#)
- TYOpenInterface
 - TYApi.h, [86](#)
- TYRegisterEventCallback
 - TYApi.h, [86](#)
- TYRegisterImuCallback
 - TYApi.h, [87](#)
- TYRemoveLogFile
 - TYApi.h, [88](#)
- TYRemoveLogServer
 - TYApi.h, [88](#)
- TYSendSoftTrigger
 - TYApi.h, [89](#)
- TYSetBool
 - TYApi.h, [90](#)
- TYSetByteArray
 - TYApi.h, [91](#)
- TYSetEnum
 - TYApi.h, [93](#)
- TYSetFloat
 - TYApi.h, [94](#)
- TYSetInt
 - TYApi.h, [96](#)
- TYSetLogLevel
 - TYApi.h, [97](#)
- TYSetLogPrefix
 - TYApi.h, [97](#)
- TYSetString
 - TYApi.h, [98](#)
- TYSetStruct
 - TYApi.h, [99](#)
- TYStartCapture
 - TYApi.h, [101](#)
- TYStopCapture
 - TYApi.h, [102](#)

TYUndistortImage
 TYImageProc.h, [136](#)
TYUpdateAllDeviceList
 TYApi.h, [103](#)
TYUpdateDeviceList
 TYApi.h, [103](#)
TYUpdateInterfaceList
 TYApi.h, [104](#)

unit_size
 TY_BYTEARRAY_ATTR, [12](#)

valid_size
 TY_BYTEARRAY_ATTR, [13](#)

z_stream_s, [33](#)