TYCamport3

3

Generated by Doxygen 1.8.17

Chapter 1

Main Page

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:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 DepthEnhenceParameters Struct Reference

Public Attributes

- float sigma_s
  filter param on space
- float sigma_r
  filter param on range
- int outlier_win_sz
  outlier filter windows ize
- float **outlier_rate**

4.1.1 Detailed Description

Definition at line 53 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 33 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 1085 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 1077 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 1071 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 1055 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 1135 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	-GAMAyz	GAMAzy
GAMAxz	1	-GAMAzx
-GAMAxy	GAMAyx	1

Definition at line 1147 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

.	.	.
SCALEx	0	0
0	SCALEy	0
0	0	SCALEz

Definition at line 1158 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 1035 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 903 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 906 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 909 of file TYDefs.h.

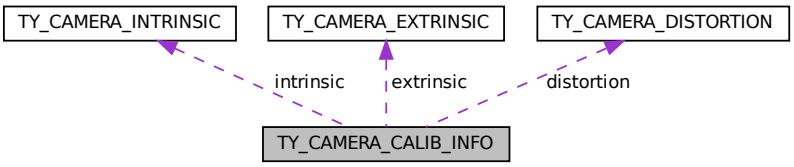
The documentation for this struct was generated from the following file:

- 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 cailbration data

See also

TYGetStruct

Definition at line 978 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 970 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 958 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 940 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 1316 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 1109 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 1201 of file TYDefs.h.

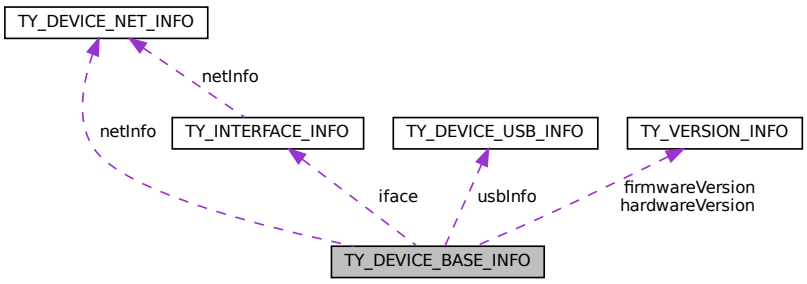
The documentation for this struct was generated from the following file:

- 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 841 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 812 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 823 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 1283 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 1260 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 914 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 1254 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 867 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

  increaing step
- float **reserved** [1]

4.29.1 Detailed Description

float range data structure

See also

TYGetFloatRange

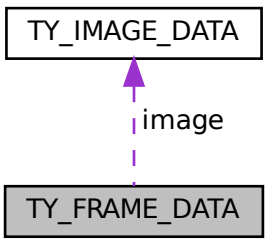
Definition at line 893 of file TYDefs.h.

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 1244 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 1167 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	-ALPHAyz	ALPHAzy
0	1	-ALPHAzx
0	0	1

Definition at line 1178 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

.	.	.
SCALEx	0	0
0	SCALEy	0
0	0	SCALEz

Definition at line 1189 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 1229 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 1118 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

  increaing step
- int32_t **reserved** [1]

4.36.1 Detailed Description

Definition at line 883 of file TYDefs.h.

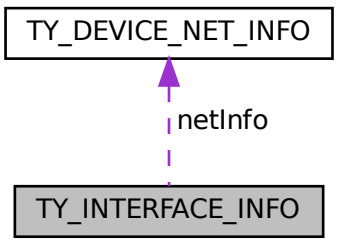
The documentation for this struct was generated from the following file:

- 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 831 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

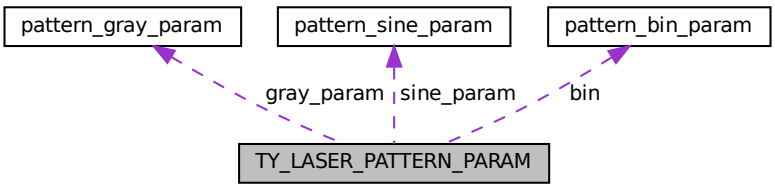
Definition at line 1219 of file TYDefs.h.

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

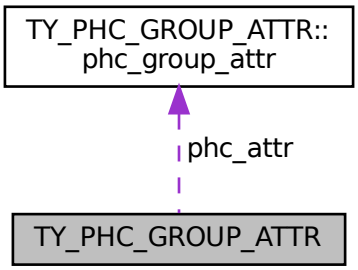
Definition at line 1091 of file TYDefs.h.

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 1051 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 1297 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 1206 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 989 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 997 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 1020 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 1028 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 921 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 803 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

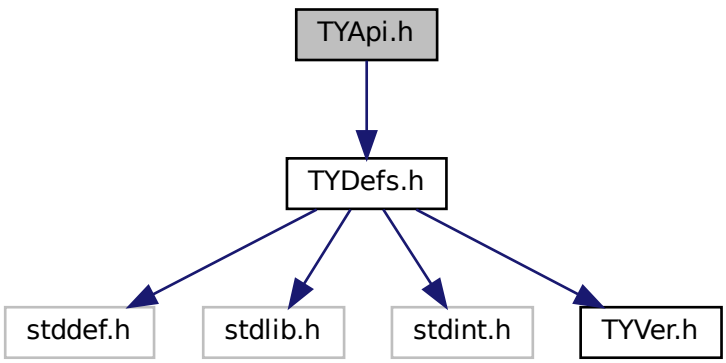
File Documentation

5.1 TYApi.h File Reference

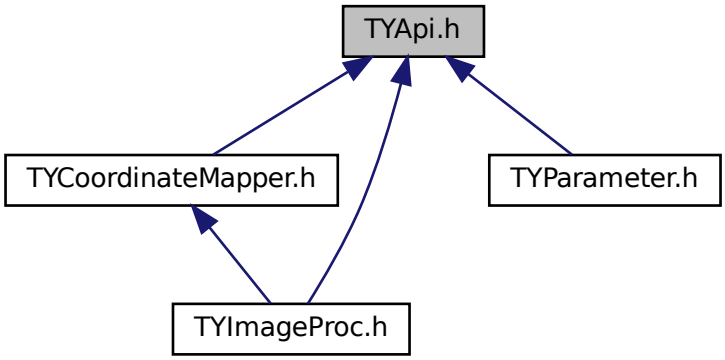
TYApi.h includes camera control and data receiving interface, which supports configuration for image resolution, frame rate, exposure
time, gain, working mode,etc.

```
#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 *filled↩ Count)

  Get interface info list.
- TY_CAPI TYHasInterface (const char *ifaceID, bool *value)

  Check if has interface.
- TY_CAPI TYOpenInterface (const char *ifaceID, TY_INTERFACE_HANDLE *outHandle)

  Open specified interface.
- TY_CAPI TYCloseInterface (TY_INTERFACE_HANDLE ifaceHandle)

  Close interface.
- TY_CAPI TYUpdateDeviceList (TY_INTERFACE_HANDLE ifaceHandle)

  Update current connected devices.
- TY_CAPI TYUpdateAllDeviceList (void)

  Update current connected devices.
- 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 *device↩ Infos, uint32_t bufferCount, uint32_t *filledDeviceCount)

  Get device info list.
- TY_CAPI TYHasDevice (TY_INTERFACE_HANDLE ifaceHandle, const char *deviceID, bool *value)

  Check whether the interface has the specified device.
- TY_CAPI TYOpenDevice (TY_INTERFACE_HANDLE ifaceHandle, const char *deviceID, TY_DEV_HANDLE *outDeviceHandle, TY_FW_ERRORCODE *outFwErrorcode=NULL)

  Open device by device ID.
- TY_CAPI TYOpenDeviceWithIP (TY_INTERFACE_HANDLE ifaceHandle, const char *IP, TY_DEV_HANDLE *deviceHandle)

  Open device by device IP, useful when a device is not listed.
- TY_CAPI TYGetDeviceInterface (TY_DEV_HANDLE hDevice, TY_INTERFACE_HANDLE *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	*filePath*	Path to the log file.
in	*lvl*	Log level.

Return values

TY_STATUS_OK	Succeed.

TY_STATUS_ERROR	Failed to add file
	`Suggestions:`
	`Please check if the file path is correct and if you have 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	*protocol*	Protocol of the server, "tcp" or "udp".
in	*ip*	IP address of the server.
in	*port*	Port of the server.
in	*lvl*	Log level.

Return values

TY_STATUS_OK	Succeed.

TY_STATUS_ERROR	Failed to add server
	`Suggestions:`
	`Please check if the ip and port are correct`

TY_STATUS_INVALID_PARAMETER	Unsupported protocol
	`Suggestions:`
	`Unsupported protocol, please use tcp or udp`

5.1.2.3 TYClearBufferQueue()

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

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

Parameters

in	*hDevice*	Device handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYClearBufferQueue called with invalid device handle Suggestions: Please check device handle Like this: TYClearBufferQueue(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 TYUpdateD
TY_STATUS_BUSY	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	*hDevice*	Device handle.
in	*reboot*	Reboot device after close.

Return values

TY_STATUS_OK	Succeed.

Return values

TY_STATUS_INVALID_HANDLE	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
TY_STATUS_TIMEOUT	Failed to close device Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the networ
TY_STATUS_DEVICE_ERROR	Failed to close device Suggestions: Possible reasons: 1.Camera device is abnormal and cannot be closed normally.
TY_STATUS_IDLE	Device has been closed.

5.1.2.5 TYCloseInterface()

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

Close interface.

Parameters

in	*ifaceHandle*	Interface to be closed.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	TYCloseInterface called with invalid interface handle Suggestions: Please check interface handle Like this: TYCloseInterface(ifaceHandle); ^ is invalid The ifaceHandle parameter you input is not recorded Possible reasons: 1.TYOpenInterface failed to open interface and get 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

TY_STATUS_OK	Succeed.

5.1.2.7 TYDisableComponents()

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

Disable components.

Parameters

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

Return values

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

Return values

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

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

Return values

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

Return values

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

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

Return values

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

Return values

TY_STATUS_WRONG_SIZE	TYEnqueueBuffer called with wrong size Suggestions: Please check your code Like this: TYEnqueueBuffer(hDevice, buffer, bufferSize); ^ is 0 or negative value
TY_STATUS_TIMEOUT	Failed to enqueue frame buffer Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the network
TY_STATUS_DEVICE_ERROR	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	*errorID*	Error id.

Return values

Error	string.

5.1.2.11 TYFetchFrame()

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

Fetch one frame.

Parameters

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

Return values

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

5.1.2.12 TYForceDeviceIP()

```
TY_CAPI TYForceDeviceIP (
            TY_INTERFACE_HANDLE ifaceHandle,
            const char * MAC,
            const char * newIP,
            const char * newNetMask,
            const char * newGateway )
```

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

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	TYForceDeviceIP called with invalid interface handle Suggestions: Please check interface handle Like this: TYForceDeviceIP(ifaceHandle, MAC, newIP, newNetMask, newGateway) ^ is invalid The ifaceHandle parameter you input is not recorded Possible reasons: 1.TYOpenInterface failed to open interface and get correct handl 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling TY

TY_STATUS_WRONG_TYPE	TYForceDeviceIP called with invalid interface type
	Suggestions:
	Please check interface type
	Usually you can get interface information by calling TYGetInterfac
	You can use TYIsNetworkInterface to check the interface type
	Only network interfaces can call TYForceDeviceIP
	Like this:
	TY_INTERFACE_INFO info; uint32_t num;
	TYGetInterfaceList(&info, 1, &num);
	if(TYIsNetworkInterface(info[0].type)) {
	TY_INTERFACE_HANDLE hIface;
	TYOpenInterface(info[0].id, &hIface);
	TYForceDeviceIP(hIface, MAC, newIP, newNetMask, newGateway);
	}

TY_STATUS_NULL_POINTER	TYForceDeviceIP called with NULL pointer Suggestions: Please check your code Like this: TYForceDeviceIP(ifaceHandle, MAC, newIP, newNetMask, newGateway) ^ or ^ or ^ or ^ is NULL
TY_STATUS_INVALID_PARAMETER	Invalid MAC address: Suggestions: Please check MAC parameter Like this: TYForceDeviceIP(ifaceHandle, MAC, newIP, newNetMask, newGateway) ^ is invalid MAC address should be six bytes of hexadecimal separated by colons For example: 00:11:22:aa:bb:cc
TY_STATUS_TIMEOUT	Failed to force set IP Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne 2.There is no camera with a matching target MAC address in the n
TY_STATUS_DEVICE_ERROR	Failed to force set IP Suggestions: Possible reasons: 1.New IP, NetMask, Gateway are incorrect, camera device refuses

5.1.2.13 TYGetBool()

```
TY_CAPI TYGetBool (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            bool * value )
```

Get value of bool feature.

Parameters

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

Return values

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

Return values

TY_STATUS_NULL_POINTER	TYGetBool called with NULL pointer Suggestions: Please check your code Like this: TYGetBool(hDevice, componentID, featureID, pValue); ^ is NULL
TY_STATUS_TIMEOUT	Failed to get bool feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get bool 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 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	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*pBuffer*	Byte buffer.
in	*bufferSize*	Size of buffer.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetByteArray called with invalid device handle Suggestions: Please check device handle Like this: TYGetByteArray(hDevice, componentID, featureID, buffer, bufferSi ^ 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

Return values

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

5.1.2.15 TYGetByteArrayAttr()

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

Return values

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

Return values

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

5.1.2.16 TYGetByteArraySize()

```
TY_CAPI TYGetByteArraySize (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            uint32_t * pSize )
```

Get the size of specified byte array zone.

Parameters

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

Return values

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

Return values

TY_STATUS_INVALID_FEATURE	Invalid feature ID Suggestions: Please check featureID parameter Like this: TYGetByteArraySize(hDevice, componentID, featureID, pSize); ^ 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
TY_STATUS_WRONG_TYPE	Feature type mismatch Suggestions: Please check the feature type Like this: TYGetByteArraySize(hDevice, componentID, featureID, pSize); ^ type mismatch The feature type you entered does not match. You can use TYFeature
TY_STATUS_NULL_POINTER	TYGetByteArraySize called with NULL pointer Suggestions: Please check your code Like this: TYGetByteArraySize(hDevice, componentID, featureID, pSize); ^ is NULL

5.1.2.17 TYGetComponentIDs()

```
TY_CAPI TYGetComponentIDs (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID * componentIDs )
```

Get all components IDs.

Parameters

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

Return values

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

Return values

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

See also

TY_DEVICE_COMPONENT_LIST

5.1.2.18 TYGetDeviceFeatureInfo()

```
TY_CAPI TYGetDeviceFeatureInfo (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_INFO * featureInfo,
            uint32_t entryCount,
            uint32_t * filledEntryCount )
```

Get the all features by comp id.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetDeviceFeatureInfo called with invalid device handle Suggestions: Please check device handle Like this: TYGetDeviceFeatureInfo(hDevice, componentID, featureInfo, entryC ^ 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

Return values

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

5.1.2.19 TYGetDeviceFeatureNumber()

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

Get the size of device features.

Parameters

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

Return values

TY_STATUS_OK	Succeed.

Return values

TY_STATUS_INVALID_HANDLE	TYGetDeviceFeatureNumber called with invalid device handle Suggestions: Please check device handle Like this: TYGetDeviceFeatureNumber(hDevice, componentID, size); ^ 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
TY_STATUS_INVALID_COMPONENT	Invalid component ID Suggestions: Please check componentID parameter Like this: TYGetDeviceFeatureNumber(hDevice, componentID, size); ^ is invalid componentID should be the value returned by TYGetComponentIDs You can also view the components of the camera by obtaining the xm
TY_STATUS_NULL_POINTER	TYGetDeviceFeatureNumber called with NULL pointer Suggestions: Please check your code Like this: TYGetDeviceFeatureNumber(hDevice, componentID, size); ^ is NULL
TY_STATUS_TIMEOUT	Failed to get feature number Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get feature number Suggestions: Possible reasons: 1.The feature of the camera device is not available or not imple 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	*hDevice*	Device handle.
out	*info*	Base info out.

Return values

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

5.1.2.21 TYGetDeviceInterface()

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

Get interface handle by device handle.

Parameters

in	*hDevice*	Device handle.
out	*pIface*	Interface handle.

Return values

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

Return values

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

5.1.2.22 TYGetDeviceList()

```
TY_CAPI TYGetDeviceList (
            TY_INTERFACE_HANDLE ifaceHandle,
            TY_DEVICE_BASE_INFO * deviceInfos,
            uint32_t bufferCount,
            uint32_t * filledDeviceCount )
```

Get device info list.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	TYGetDeviceList called with invalid interface handle Suggestions: Please check interface handle Like this: TYGetDeviceList(ifaceHandle, pDeviceInfos, bufferCount, pFilledDe ^ 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
TY_STATUS_NULL_POINTER	TYGetDeviceList called with NULL pointer Suggestions: Please check your code Like this: TYGetDeviceList(ifaceHandle, pDeviceInfos, bufferCount, pFilledCo ^ is NULL or ^ is 0 or ^ is NULL

5.1.2.23 TYGetDeviceNumber()

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

Get number of current connected devices.

Parameters

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

Return values

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

5.1.2.24 TYGetDeviceXML()

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

Get the Device xml string.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	Not call TYInitLib

TY_STATUS_WRONG_SIZE	XML buffer size is not enough
	`Suggestions:`
	`  XML buffer size is not enough`
	`  Like this:`
	`  TYGetDeviceXML(hDevice, xml, in_size, out_size);`
	`  ^ is invalid`
	`  XML buffer size is not enough, please use TYGetDeviceXMLSize to get th`

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

TY_STATUS_NULL_POINTER	TYGetDeviceXML called with NULL pointer
	`Suggestions:`
	`  Please check your code`
	`  Like this:`
	`  TYGetDeviceXML(hDevice, xml, in_size, out_size);`
	`  ^ or ^ is NULL`

5.1.2.25 TYGetDeviceXMLSize()

```
TY_CAPI TYGetDeviceXMLSize (
            TY_DEV_HANDLE hDevice,
            uint32_t * size )
```

Get the Device xml size.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	Not call TYInitLib

Return values

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

5.1.2.26 TYGetEnabledComponents()

```
TY_CAPI TYGetEnabledComponents (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID * componentIDs )
```

Get all enabled components IDs.

Parameters

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

Return values

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

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

Return values

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

Return values

TY_STATUS_WRONG_TYPE	Feature type mismatch
	`Suggestions:`
	`  Please check the feature type`
	`  Like this:`
	`  TYGetEnum(hDevice, componentID, featureID, pValue);`
	`  ^ type mismatch`
	`  The feature type you entered does not match. You can use TYFeature`

TY_STATUS_NULL_POINTER	TYGetEnum called with NULL pointer
	`Suggestions:`
	`  Please check your code`
	`  Like this:`
	`  TYGetEnum(hDevice, componentID, featureID, pValue);`
	`  ^ is NULL`

TY_STATUS_TIMEOUT	Failed to get enum feature
	`Suggestions:`
	`  Possible reasons:`
	`  1.Network communication is abnormal, please check whether the ne`

TY_STATUS_DEVICE_ERROR	Failed to get enum feature
	`Suggestions:`
	`  Possible reasons:`
	`  1.The feature of the camera device is not available or not imple`
	`  2.Camera device is abnormal and cannot get enum feature`

5.1.2.28 TYGetEnumEntryCount()

```
TY_CAPI TYGetEnumEntryCount (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            uint32_t * entryCount )
```

Get number of enum entries.

Parameters

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

Return values

TY_STATUS_OK	Succeed.

Return values

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

5.1.2.29 TYGetEnumEntryInfo()

```
TY_CAPI TYGetEnumEntryInfo (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            TY_ENUM_ENTRY * entries,
            uint32_t entryCount,
            uint32_t * filledEntryCount )
```

Get list of enum entries.

Parameters

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

Return values

TY_STATUS_OK	Succeed.

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

TY_STATUS_INVALID_COMPONENT	Invalid component ID: d
	`Suggestions:`
	`  Please check componentID parameter`
	`  Like this:`
	`  TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, ent`
	`  ^ is invalid`
	`  componentID should be the value returned by TYGetComponentIDs`
	`  You can also view the components of the camera by obtaining the xm`

TY_STATUS_INVALID_FEATURE	Invalid feature ID: d
	`Suggestions:`
	`  Please check featureID parameter`
	`  Like this:`
	`  TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, ent`
	`  ^ is invalid`
	`  You entered an invalid featureID parameter`
	`  You can get a list of features of the camera device through TYGetE`
	`  You can also view the features of the camera device by obtaining t`

TY_STATUS_WRONG_TYPE	Feature type mismatch
	`Suggestions:`
	`  Please check the feature type`
	`  Like this:`
	`  TYGetEnumEntryInfo(hDevice, componentID, featureID, entries, ent`
	`  ^ type mismatch`
	`  The feature type you entered does not match. You can use TYFeature`

TY_STATUS_NULL_POINTER	TYGetEnumEntryInfo called with NULL pointer
	`Suggestions:`
	`  Please check your code`
	`  Like this:`
	`  TYGetEnumEntryInfo(hDevice, componentID, featureID, pEnumDescrip`
	`  ^`

5.1.2.30 TYGetFeatureInfo()

```
TY_CAPI TYGetFeatureInfo (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            TY_FEATURE_INFO * featureInfo )
```

Get feature info.

Parameters

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

Return values

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

5.1.2.31 TYGetFloat()

```
TY_CAPI TYGetFloat (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            float * value )
```

Get value of float feature.

Parameters

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

Return values

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

Return values

TY_STATUS_NULL_POINTER	TYGetFloat called with NULL pointer Suggestions: Please check your code Like this: TYGetFloat(hDevice, componentID, featureID, pValue); ^ is NULL
TY_STATUS_TIMEOUT	Failed to get float feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get float 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 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	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*floatRange*	Float range to be filled.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYGetFloatRange called with invalid device handle Suggestions: Please check device handle Like this: TYGetFloatRange(hDevice, componentID, featureID, pFloatRange); ^ 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

Return values

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

5.1.2.33 TYGetFrameBufferSize()

```
TY_CAPI TYGetFrameBufferSize (
            TY_DEV_HANDLE hDevice,
            uint32_t * bufferSize )
```

Get total buffer size of one frame in current configuration.

Parameters

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

Return values

TY_STATUS_OK	Succeed.

Return values

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

5.1.2.34 TYGetInt()

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

Return values

TY_STATUS_OK	Succeed.

Return values

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

5.1.2.35 TYGetInterfaceList()

```
TY_CAPI TYGetInterfaceList (
            TY_INTERFACE_INFO * pIfaceInfos,
            uint32_t bufferCount,
            uint32_t * filledCount )
```

Get interface info list.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

TY_STATUS_NULL_POINTER	TYGetInterfaceList called with NULL pointer
	`Suggestions:`
	`  Please check your code`
	`  Like this:`
	`  TYGetInterfaceList(pIfaceInfos, bufferCount, filledCount);`
	`  ^ or ^ is NULL`

5.1.2.36 TYGetInterfaceNumber()

```
TY_CAPI TYGetInterfaceNumber (
            uint32_t * pNumIfaces )
```

Get number of current interfaces.

Parameters

out	*pNumIfaces*	Number of interfaces.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

TY_STATUS_NULL_POINTER	TYGetInterfaceNumber called with NULL pointer
	`Suggestions:`
	`  Please check your code`
	`  Like this:`
	`  TYGetInterfaceNumber(pNumIfaces);`
	`  ^ is NULL`

5.1.2.37 TYGetIntRange()

```
TY_CAPI TYGetIntRange (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            TY_INT_RANGE * intRange )
```

Get value range of integer feature.

Parameters

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

Return values

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

Return values

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

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

Return values

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

Return values

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

See also

TYGetStringLength

5.1.2.39 TYGetStringLength()

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

Return values

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

See also

TYGetString

5.1.2.40 TYGetStruct()

```
TY_CAPI TYGetStruct (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            void * pStruct,
            uint32_t structSize )
```

Get value of struct.

Parameters

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

Return values

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

Return values

TY_STATUS_NULL_POINTER	TYGetStruct called with NULL pointer Suggestions: Please check your code Like this: TYGetStruct(hDevice, componentID, featureID, pStruct, structSize ^ is NULL
TY_STATUS_WRONG_SIZE	Struct size mismatch Suggestions: Please check the struct size Like this: TYGetStruct(hDevice, componentID, featureID, pStruct, structSize ^ is inval The struct size you entered does not match
TY_STATUS_TIMEOUT	Failed to get struct feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to get struct 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 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	*ifaceHandle*	Interface handle.
in	*deviceID*	Device ID string, can be get from TY_DEVICE_BASE_INFO.
out	*value*	True if the device exists.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

Return values

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

5.1.2.42 TYHasFeature()

```
TY_CAPI TYHasFeature (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            bool * value )
```

Check whether a component has a specific feature.

Parameters

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

Return values

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

Return values

TY_STATUS_INVALID_COMPONENT	Invalid component ID
	`Suggestions:`
	`  Please check componentID parameter`
	`  Like this:`
	`  TYHasFeature(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 xm`

TY_STATUS_NULL_POINTER	TYHasFeature called with NULL pointer
	`Suggestions:`
	`  Please check your code`
	`  Like this:`
	`  TYHasFeature(hDevice, componentID, featureID, value);`
	`  ^ is NULL`

5.1.2.43 TYHasInterface()

```
TY_CAPI TYHasInterface (
            const char * ifaceID,
            bool * value )
```

Check if has interface.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

TY_STATUS_NULL_POINTER	TYHasInterface called with NULL pointer
	`Suggestions:`
	`  Please check your code`
	`  Like this:`
	`  TYHasInterface(ifaceID, value);`
	`  ^ is NULL`

See also

TYGetInterfaceList

5.1.2.44 TYLibVersion()

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

Get current library version.

Parameters

out	*version*	Version infomation to be filled.

Return values

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

5.1.2.45 TYOpenDevice()

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

Return values

TY_STATUS_INVALID_INTERFACE	TYOpenDevice called with invalid interface handle Suggestions: Please check interface handle Like this: TYOpenDevice(ifaceHandle, deviceID, outDeviceHandle, outFwErrorc ^ is invalid The ifaceHandle parameter you input is not recorded Possible reasons: 1.TYOpenInterface failed to open interface and get correct handl 2.Memory in stack to store handle data is corrupted 3.After getting handle, you updated interface list by calling TY
TY_STATUS_NULL_POINTER	TYOpenDevice called with NULL pointer Suggestions: Please check your code Like this: TYOpenDevice(ifaceHandle, deviceID, outDeviceHandle, outFwErrorc ^ or ^ is NULL
TY_STATUS_INVALID_PARAMETER	TYOpenDevice called with invalid device ID: s Suggestions: Please check deviceID parameter Like this: TYOpenDevice(ifaceHandle, deviceID, outDeviceHandle, outFwErrorc ^ 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
TY_STATUS_BUSY	Failed to open device Suggestions: Possible reasons: 1.Camera is occupied, please check if other processes on this ma or other host machines are occupying the camera. If the camera 2.A third-party program is written into the camera, please conta

TY_STATUS_FIRMWARE_ERROR	Device opened successfully, but firmware error code is not 0
	Suggestions:
	Some functions of the device may have exceptions, please check the
	TY_FW_ERRORCODE outFwErrorcode;
	TYOpenDevice(ifaceHandle, deviceID, outDeviceHandle, &outFwErrorco
	if(outFwErrorcode != 0) {
	parse_firmware_errcode(outFwErrorcode);
	}

TY_STATUS_DEVICE_ERROR	Failed to open device Suggestions: Possible reasons: 1.A third-party program is written into the camera, please conta 2.The camera IP address is not in the same network segment as th If the camera IP address is not in the same network segment as If there is a routing connection between your host and the cam Otherwise, you can modify the camera IP address or the host IP If you need to modify the camera IP address, please refer to S 3.Network communication is abnormal, please check whether the ne

5.1.2.46 TYOpenDeviceWithIP()

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

Return values

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

Return values

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

5.1.2.47 TYOpenInterface()

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

Open specified interface.

Parameters

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

Return values

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

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

Return values

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

5.1.2.49 TYRegisterImuCallback()

```
TY_CAPI TYRegisterImuCallback (
            TY_DEV_HANDLE hDevice,
            TY_IMU_CALLBACK callback,
            void * userdata )
```

Register imu callback. Register NULL to clean callback.

Parameters

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

Return values

TY_STATUS_OK	Succeed.

Return values

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

5.1.2.50 TYRemoveLogFile()

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

Remove log file.

Parameters

in	*filePath*	Path to the log file.

Return values

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

5.1.2.51 TYRemoveLogServer()

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

Remove log server.

Parameters

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

Return values

TY_STATUS_OK	Succeed.

TY_STATUS_ERROR	Failed to remove server
	`Suggestions:`
	`  Please check if the ip and port are correct`

TY_STATUS_INVALID_PARAMETER	Unsupported protocol
	`Suggestions:`
	`  Unsupported protocol, please use tcp or udp`

5.1.2.52 TYSendSoftTrigger()

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

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

Parameters

in	*hDevice*	Device handle.

Return values

TY_STATUS_OK	Succeed.

TY_STATUS_INVALID_HANDLE	TYSendSoftTrigger called with invalid device handle
	`Suggestions:`
	`  Please check device handle`
	`  Like this:`
	`  TYSendSoftTrigger(hDevice);`
	`  ^ is invalid`
	`  The hDevice parameter you input is not recorded`
	`  Possible reasons:`
	`  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 TYUpdate`

TY_STATUS_INVALID_FEATURE	Not support soft trigger.

TY_STATUS_IDLE	Camera device is not started
	`Suggestions:`
	`  Please start the camera device first`
	`  Like this:`
	`  TYStartCapture(hDevice);`
	`  TYSendSoftTrigger(hDevice);`

TY_STATUS_WRONG_MODE	Not in trigger mode.

TY_STATUS_DEVICE_ERROR	Failed to send soft trigger
	`Suggestions:`
	`  Possible reasons:`
	`  1.Camera device is abnormal and cannot send soft trigger.`
	`  2.Network communication is abnormal, please check whether the netwo`

Return values

TY_STATUS_BUSY	Failed to send soft trigger Suggestions: Possible reasons: 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	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
in	*value*	Bool value.

Return values

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

Return values

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

5.1.2.54 TYSetByteArray()

```
TY_CAPI TYSetByteArray (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            const uint8_t * pBuffer,
            uint32_t bufferSize )
```

Write byte array to device.

Parameters

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

Return values

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

Return values

TY_STATUS_DEVICE_ERROR	Failed to set byte array feature Suggestions: Possible reasons: 1.The feature of the camera device is not available or not imple 2.Camera device is abnormal and cannot 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	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
in	*value*	Enum value.

Return values

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

Return values

TY_STATUS_INVALID_FEATURE	Invalid feature ID
	`Suggestions:`
	`  Please check featureID parameter`
	`  Like this:`
	`  TYSetEnum(hDevice, componentID, featureID, value);`
	`  ^ is invalid`
	`  You entered an invalid featureID parameter`
	`  You can get a list of features of the camera device through TYGetI`
	`  You can also view the features of the camera device by obtaining t`

TY_STATUS_NOT_PERMITTED	The feature is not writable.

TY_STATUS_WRONG_TYPE	Feature type mismatch
	`Suggestions:`
	`  Please check the feature type`
	`  Like this:`
	`  TYSetEnum(hDevice, componentID, featureID, value);`
	`  ^ type mismatch`
	`  The feature type you entered does not match. You can use TYFeature`

TY_STATUS_INVALID_PARAMETER	Out of range
	`Suggestions:`
	`  Please check the value`
	`  Like this:`
	`  TYSetEnum(hDevice, componentID, featureID, value);`
	`  ^ is out of range`
	`  The value is out of range, please use TYGetEnumEntryInfo to get th`

TY_STATUS_BUSY	Device is capturing, the feature is locked.

TY_STATUS_TIMEOUT	Failed to set enum feature
	`Suggestions:`
	`  Possible reasons:`
	`  1.Network communication is abnormal, please check whether the ne`

TY_STATUS_DEVICE_ERROR	Failed to set enum feature
	`Suggestions:`
	`  Possible reasons:`
	`  1.The feature of the camera device is not available or not imple`
	`  2.Camera device is abnormal and cannot set enum feature`

5.1.2.56 TYSetFloat()

```
TY_CAPI TYSetFloat (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            float value )
```

Set value of float feature.

Parameters

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

Return values

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

5.1.2.57 TYSetInt()

```
TY_CAPI TYSetInt (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            int32_t value )
```

Set value of integer feature.

Parameters

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

Return values

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

Return values

TY_STATUS_WRONG_TYPE	Feature type mismatch
	`Suggestions:`
	`  Please check the feature type`
	`  Like this:`
	`  TYSetInt(hDevice, componentID, featureID, value);`
	`  ^ type mismatch`
	`  The feature type you entered does not match. You can use TYFeature`

TY_STATUS_OUT_OF_RANGE	Out of range
	`Suggestions:`
	`  Please check the value`
	`  Like this:`
	`  TYSetInt(hDevice, componentID, featureID, value);`
	`  ^ is out of range`
	`  The value is out of range, please use TYGetIntRange to get the ra`

TY_STATUS_BUSY	Device is capturing, the feature is locked.

TY_STATUS_TIMEOUT	Failed to set int feature
	`Suggestions:`
	`  Possible reasons:`
	`  1.Network communication is abnormal, please check whether the n`

TY_STATUS_DEVICE_ERROR	Failed to set int feature
	`Suggestions:`
	`  Possible reasons:`
	`  1.The feature of the camera device is not available or not impl`
	`  2.Camera device is abnormal and cannot set int feature`

5.1.2.58 TYSetLogLevel()

```
TY_CAPI TYSetLogLevel (
            TY_LOG_LEVEL lvl )
```

Set log level.

Parameters

in	*lvl*	Log level.

Return values

TY_STATUS_OK	Succeed.

5.1.2.59 TYSetLogPrefix()

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

set log prefix

Parameters

in	*prefix*	Prefix string.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_PARAMETER	Prefix is empty or prefix is too long Suggestions: Prefix is empty or prefix is too long, cannot be set Like this: TYSetLogPrefix(prefix); ^ prefix is empty or prefix is too 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	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
in	*buffer*	String buffer.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYSetString called with invalid device handle Suggestions: Please check device handle Like this: TYSetString(hDevice, componentID, featureID, pBuffer); ^ 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

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

5.1.2.61 TYSetStruct()

```
TY_CAPI TYSetStruct (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
```

```
    void * pStruct,
    uint32_t structSize )
```

Set value of struct.

Parameters

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

Return values

TY_STATUS_OK	Succeed.

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

TY_STATUS_INVALID_COMPONENT	Invalid component ID
	`Suggestions:`
	`  Please check componentID parameter`
	`  Like this:`
	`  TYSetStruct(hDevice, componentID, featureID, pStruct, structSize`
	`  ^ is invalid`
	`  componentID should be the value returned by TYGetComponentIDs`
	`  You can also view the components of the camera by obtaining the x`

TY_STATUS_INVALID_FEATURE	Invalid feature ID
	`Suggestions:`
	`  Please check featureID parameter`
	`  Like this:`
	`  TYSetStruct(hDevice, componentID, featureID, pStruct, structSize`
	`  ^ is invalid`
	`  You entered an invalid featureID parameter`
	`  You can get a list of features of the camera device through TYGet`
	`  You can also view the features of the camera device by obtaining`

TY_STATUS_NOT_PERMITTED	The feature is not writable.

TY_STATUS_WRONG_TYPE	Feature type mismatch
	`Suggestions:`
	`  Please check the feature type`
	`  Like this:`
	`  TYSetStruct(hDevice, componentID, featureID, pStruct, structSize`
	`  ^ type mismatch`
	`  The feature type you entered does not match. You can use TYFeature`

Return values

TY_STATUS_NULL_POINTER	TYSetStruct called with NULL pointer Suggestions: Please check your code Like this: TYSetStruct(hDevice, componentID, featureID, pStruct, structSize ^ is NULL
TY_STATUS_WRONG_SIZE	Struct size mismatch Suggestions: Please check the struct size Like this: TYSetStruct(hDevice, componentID, featureID, pStruct, structSize ^ is inval The struct size you entered does not match
TY_STATUS_BUSY	Device is capturing, the feature is locked.
TY_STATUS_TIMEOUT	Failed to set struct feature Suggestions: Possible reasons: 1.Network communication is abnormal, please check whether the ne
TY_STATUS_DEVICE_ERROR	Failed to set struct feature Suggestions: Possible reasons: 1.The feature of the camera device is not available or not imple 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	*hDevice*	Device handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	TYStartCapture called with invalid device handle Suggestions: Please check device handle Like this: TYStartCapture(hDevice); ^ is invalid The hDevice parameter you input is not recorded Possible reasons: 1.TYOpenDevice failed to open device and get 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

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

5.1.2.63 TYStopCapture()

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

Stop capture.

Parameters

in	*hDevice*	Device handle.

Return values

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

Return values

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

5.1.2.64 TYUpdateAllDeviceList()

```
TY_CAPI TYUpdateAllDeviceList (
            void  )
```

Update current connected devices.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

5.1.2.65 TYUpdateDeviceList()

```
TY_CAPI TYUpdateDeviceList (
            TY_INTERFACE_HANDLE ifaceHandle )
```

Update current connected devices.

Parameters

in	*ifaceHandle*	Interface handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

Return values

TY_STATUS_INVALID_INTERFACE	TYUpdateDeviceList called with invalid interface handle Suggestions: Please check interface handle Like this: TYUpdateDeviceList(ifaceHandle); ^ is invalid The ifaceHandle parameter you input is not recorded Possible reasons: 1.TYOpenInterface failed to open interface and get 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

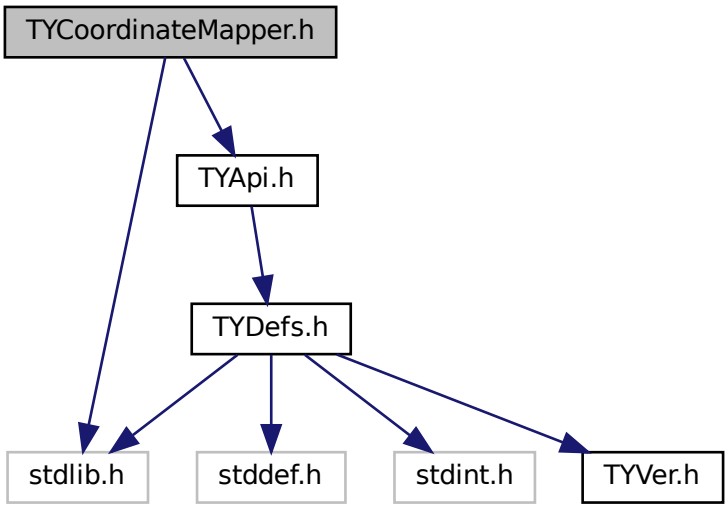
TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

5.2 TYCoordinateMapper.h File Reference

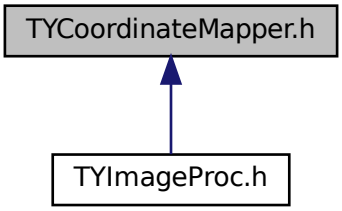
Coordinate Conversion API.

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

Include dependency graph for TYCoordinateMapper.h:



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



Classes

- struct TY_PIXEL_DESC
- struct TY_PIXEL_COLOR_DESC

Macros

- #define **TYMAP_CHECKRET**(f, bufToFree)

Typedefs

- typedef struct TY_PIXEL_DESC **TY_PIXEL_DESC**
- typedef struct TY_PIXEL_COLOR_DESC **TY_PIXEL_COLOR_DESC**

Functions

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

  Calculate 4x4 extrinsic matrix's inverse matrix.
- TY_CAPI TYMapDepthToPoint3d (const TY_CAMERA_CALIB_INFO *src_calib, uint32_t depthW, uint32_t depthH, const TY_PIXEL_DESC *depthPixels, uint32_t count, TY_VECT_3F *point3d, float f_scale_unit=1.0f)

  Map pixels on depth image to 3D points.
- TY_CAPI TYMapPoint3dToDepth (const TY_CAMERA_CALIB_INFO *dst_calib, const TY_VECT_3F *point3d, uint32_t count, uint32_t depthW, uint32_t depthH, TY_PIXEL_DESC *depth, float f_scale_unit=1.0f)

  Map 3D points to pixels on depth image. Reverse operation of TYMapDepthToPoint3d.
- TY_CAPI TYMapDepthImageToPoint3d (const TY_CAMERA_CALIB_INFO *src_calib, int32_t imageW, int32_t imageH, const uint16_t *depth, TY_VECT_3F *point3d, float f_scale_unit=1.0f)

  Map depth image to 3D points. 0 depth pixels maps to (NAN, NAN, NAN).
- TY_CAPI TYDepthImageFillEmptyRegion (uint16_t *depth, uint32_t depthW, uint32_t depthH)

  Fill depth image empty region.
- TY_CAPI TYMapPoint3dToDepthImage (const TY_CAMERA_CALIB_INFO *dst_calib, const TY_VECT_3F *point3d, uint32_t count, uint32_t depthW, uint32_t depthH, uint16_t *depth, float f_target_scale=1.0f)

  Map 3D points to depth image. (NAN, NAN, NAN) will be skipped.
- TY_CAPI TYMapPoint3dToPoint3d (const TY_CAMERA_EXTRINSIC *extrinsic, const TY_VECT_3F *point3dFrom, int32_t count, TY_VECT_3F *point3dTo)

  Map 3D points to another coordinate.

5.2.1 Detailed Description

Coordinate Conversion API.

Note

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

Copyright

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	*depth*	Depth image pixels.
in	*depthW*	Width of current depth image.
in	*depthH*	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	*orgExtrinsic*	Input extrinsic matrix.
out	*invExtrinsic*	Inverse matrix.

Return values

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

Return values

TY_STATUS_OK	Succeed.

5.2.3.4 TYMapDepthToPoint3d()

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

Parameters

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

Return values

TY_STATUS_OK	Succeed.

5.2.3.5 TYMapPoint3dToDepth()

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

Return values

TY_STATUS_OK	Succeed.

5.2.3.6 TYMapPoint3dToDepthImage()

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

Return values

TY_STATUS_OK	Succeed.

5.2.3.7 TYMapPoint3dToPoint3d()

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

Return values

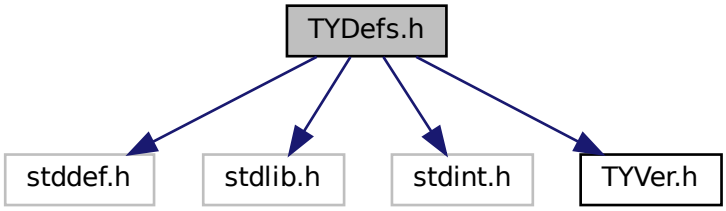
TY_STATUS_OK	Succeed.

5.3 TYDefs.h File Reference

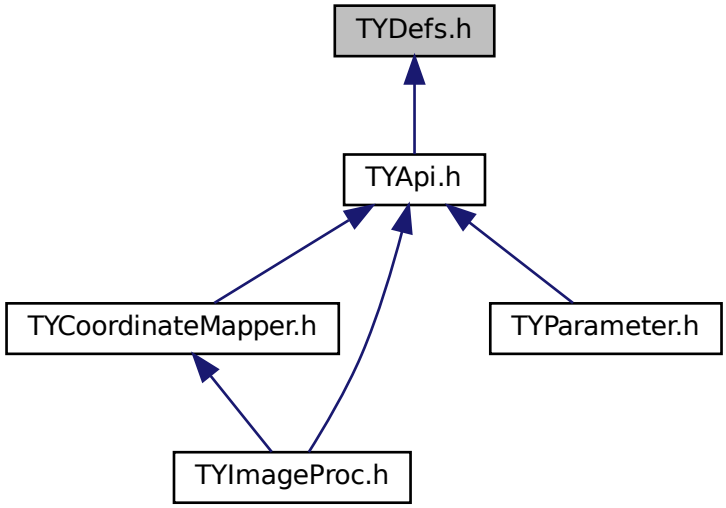
TYDefs.h includes camera control and data receiving data definitions which supports configuration for image resolution, frame rate, exposure
time, gain, working mode,etc.

```
#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 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_INVALID_PARAMETER** = -1006, **TY_STATUS_INVALID_HANDLE** = -1007,
  TY_STATUS_INVALID_COMPONENT = -1008, **TY_STATUS_INVALID_FEATURE** = -1009, **TY_STATUS_WRONG_TYPE** = -1010, **TY_STATUS_WRONG_SIZE** = -1011,
  TY_STATUS_OUT_OF_MEMORY = -1012, **TY_STATUS_OUT_OF_RANGE** = -1013, **TY_STATUS_TIMEOUT** = -1014, **TY_STATUS_WRONG_MODE** = -1015,
  TY_STATUS_BUSY = -1016, **TY_STATUS_IDLE** = -1017, **TY_STATUS_NO_DATA** = -1018, **TY_STATUS_NO_BUFFER** = -1019,
  TY_STATUS_NULL_POINTER = -1020, **TY_STATUS_READONLY_FEATURE** = -1021, **TY_STATUS_INVALID_DESCRIPTOR** = -1022, **TY_STATUS_INVALID_INTERFACE** = -1023,
  TY_STATUS_FIRMWARE_ERROR = -1024, **TY_STATUS_ERROR_XML** = -1025, **TY_STATUS_DEV_EPERM** = -1, **TY_STATUS_DEV_EIO** = -5,
  TY_STATUS_DEV_ENOMEM = -12, **TY_STATUS_DEV_EBUSY** = -16, **TY_STATUS_DEV_EINVAL** = -22
  }

  API call return status.
- enum **TY_FW_ERRORCODE_LIST** : uint32_t {
  TY_FW_ERRORCODE_CAM0_NOT_DETECTED = 0x00000001, **TY_FW_ERRORCODE_CAM1_NOT_DETECTED** = 0x00000002, **TY_FW_ERRORCODE_CAM2_NOT_DETECTED** = 0x00000004, **TY_FW_ERRORCODE_POE_NOT_INIT** = 0x00000008,
  TY_FW_ERRORCODE_RECMAP_NOT_CORRECT = 0x00000010, **TY_FW_ERRORCODE_LOOKUPTABLE_NOT_CORRECT** = 0x00000020, **TY_FW_ERRORCODE_DRV8899_NOT_INIT** = 0x00000040, **TY_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_ERRORCODE_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_ERROR** = -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_LE
  = 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_CMD = 0x0000, **TY_FEATURE_INT** = 0x1000, **TY_FEATURE_FLOAT** = 0X2000, **TY_FE↩ ATURE_ENUM** = 0x3000,
  TY_FEATURE_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_ENUM_USER_SET_DEFAULT** = 0x0301 | TY_FEATURE_ENUM, TY_ENUM_USER_SET_CURRENT = 0x0304 | TY_FEATURE_ENUM,
  TY_ENUM_USER_SET_SELECTOR = 0x0302 | TY_FEATURE_ENUM,
  TY_CMD_USER_SET_LOAD = 0x0305 | TY_FEATURE_CMD, TY_CMD_USER_SET_SAVE = 0x0303 | TY_FEATURE_CMD, TY_STRING_USER_SET_DESCRIPTION = 0x0306 | TY_FEATURE_STRING,
  TY_STRUCT_TRIGGER_PARAM = 0x0523 | TY_FEATURE_STRUCT,
  TY_STRUCT_TRIGGER_PARAM_EX = 0x0525 | TY_FEATURE_STRUCT, TY_STRUCT_TRIGGER_TIMER_LIST = 0x0526 | TY_FEATURE_STRUCT, TY_STRUCT_TRIGGER_TIMER_PERIOD = 0x0527 | TY_FEATUR↩ E_STRUCT, TY_BOOL_KEEP_ALIVE_ONOFF = 0x0203 | TY_FEATURE_BOOL,
  TY_INT_KEEP_ALIVE_TIMEOUT = 0x0204 | TY_FEATURE_INT, TY_BOOL_CMOS_SYNC = 0x0205 | T↩ Y_FEATURE_BOOL, TY_INT_TRIGGER_DELAY_US = 0x0206 | TY_FEATURE_INT, TY_BOOL_TRIGGER_OUT_IO = 0x0207 | TY_FEATURE_BOOL,
  TY_INT_TRIGGER_DURATION_US = 0x0208 | TY_FEATURE_INT, TY_ENUM_STREAM_ASYNC = 0x0209 | TY_FEATURE_ENUM, TY_INT_CAPTURE_TIME_US = 0x0210 | TY_FEATURE_INT, TY_ENUM_TIME_SYNC_TYPE = 0x0211 | TY_FEATURE_ENUM,
  TY_BOOL_TIME_SYNC_READY = 0x0212 | TY_FEATURE_BOOL, TY_BOOL_IR_FLASHLIGHT = 0x0213 | TY_FEATURE_BOOL, TY_INT_IR_FLASHLIGHT_INTENSITY = 0x0214 | TY_FEATURE_INT, TY_STRUCT_DO0_WORKMODE = 0x0215 | TY_FEATURE_STRUCT,
  TY_STRUCT_DI0_WORKMODE = 0x0216 | TY_FEATURE_STRUCT, TY_STRUCT_DO1_WORKMODE = 0x0217 | TY_FEATURE_STRUCT, TY_STRUCT_DI1_WORKMODE = 0x0218 | TY_FEATURE_STRUCT, TY_STRUCT_DO2_WORKMODE = 0x0219 | TY_FEATURE_STRUCT,
  TY_STRUCT_DI2_WORKMODE = 0x0220 | TY_FEATURE_STRUCT, TY_BOOL_RGB_FLASHLIGHT = 0x0221 | TY_FEATURE_BOOL, TY_INT_RGB_FLASHLIGHT_INTENSITY = 0x0222 | TY_FEATURE_INT, **TY_ENUM_CONFIG_MODE** = 0x0221 | TY_FEATURE_ENUM,
  TY_ENUM_TEMPERATURE_ID = 0x0223 | TY_FEATURE_ENUM, **TY_STRUCT_TEMPERATURE** = 0x0224 | TY_FEATURE_STRUCT, TY_BOOL_AUTO_EXPOSURE = 0x0300 | TY_FEATURE_BOOL, TY_INT_EXPOSURE_TIME = 0x0301 | TY_FEATURE_INT,
  TY_BOOL_AUTO_GAIN = 0x0302 | TY_FEATURE_BOOL, TY_INT_GAIN = 0x0303 | TY_FEATURE_INT, TY_BOOL_AUTO_AWB = 0x0304 | TY_FEATURE_BOOL, TY_STRUCT_AEC_ROI = 0x0305 | TY_FEA↩ TURE_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_LASER_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_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_FEATURE_INT,
TY_INT_SGBM_TEXTURE_OFFSET = 0x062F | TY_FEATURE_INT, TY_INT_SGBM_TEXTURE_THRESH = 0x0630 | TY_FEATURE_INT, TY_STRUCT_PHC_GROUP_ATTR = 0x0710 | TY_FEATURE_STRUCT,
TY_ENUM_DEPTH_QUALITY = 0x0900 | TY_FEATURE_ENUM,
TY_INT_FILTER_THRESHOLD = 0x0901 | TY_FEATURE_INT, TY_INT_TOF_CHANNEL = 0x0902 | TY_FEATURE_INT, TY_INT_TOF_MODULATION_THRESHOLD = 0x0903 | TY_FEATURE_INT,

TY_STRUCT_TOF_FREQ = 0x0904 | TY_FEATURE_STRUCT,
TY_BOOL_TOF_ANTI_INTERFERENCE = 0x0905 | TY_FEATURE_BOOL, TY_INT_TOF_ANTI_SUNLIGHT_INDEX = 0x0906 | TY_FEATURE_INT, TY_INT_MAX_SPECKLE_SIZE = 0x0907 | TY_FEATURE_INT, TY_INT_MAX_SPECKLE_DIFF = 0x0908 | TY_FEATURE_INT }

feature for component definitions

- enum **TY_CONFIG_MODE_LIST** : uint32_t {
TY_CONFIG_MODE_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_QUALIT↩Y_MEDIUM** = 2, **TY_DEPTH_QUALITY_HIGH** = 4 }
- enum TY_TRIGGER_POL_LIST : uint32_t { **TY_TRIGGER_POL_FALLINGEDGE** = 0, **TY_TRIGGER_P↩OL_RISINGEDGE** = 1 }

set external trigger signal edge

- enum TY_INTERFACE_TYPE_LIST : uint32_t {
TY_INTERFACE_UNKNOWN = 0, **TY_INTERFACE_RAW** = 1, **TY_INTERFACE_USB** = 2, **TY_INTERF↩ACE_ETHERNET** = 4,
TY_INTERFACE_IEEE80211 = 8, **TY_INTERFACE_ALL** = 0xffff }
- enum TY_ACCESS_MODE_LIST : uint32_t { **TY_ACCESS_READABLE** = 0x1, **TY_ACCESS_WRITABLE** = 0x2 }
- enum TY_STREAM_ASYNC_MODE_LIST : uint32_t {
TY_STREAM_ASYNC_OFF = 0, **TY_STREAM_ASYNC_DEPTH** = 1, **TY_STREAM_ASYNC_RGB** = 2, **T↩Y_STREAM_ASYNC_DEPTH_RGB** = 3,
TY_STREAM_ASYNC_ALL = 0xff }

stream async mode

- enum TYPixFmtList : uint32_t {
TYPixelFormatMono8 = 0x01080001, **TYPixelFormatBayerGBRG8** = 0x0108000A, **TYPixelFormat↩BayerBGGR8** = 0x0108000B, **TYPixelFormatBayerGRBG8** = 0x01080008,
TYPixelFormatBayerRGGB8 = 0x01080009, **TYPixelFormatPacketMono10** = 0x010A0046, **TYPixel↩FormatPacketBayerGBRG10** = 0x010A0054, **TYPixelFormatPacketBayerBGGR10** = 0x010A0052,
TYPixelFormatPacketBayerGRBG10 = 0x010A0056, **TYPixelFormatPacketBayerRGGB10** = 0x010↩A0058, **TYPixelFormatPacketMono12** = 0x010C0047, **TYPixelFormatPacketBayerGBRG12** = 0x010↩C0055,
TYPixelFormatPacketBayerBGGR12 = 0x010C0053, **TYPixelFormatPacketBayerGRBG12** = 0x010↩C0057, **TYPixelFormatPacketBayerRGGB12** = 0x010C0059, **TYPixelFormatMono10** = 0x810A0046,
TYPixelFormatBayerGBRG10 = 0x810A0054, **TYPixelFormatBayerBGGR10** = 0x810A0052, **TYPixel↩FormatBayerGRBG10** = 0x810A0056, **TYPixelFormatBayerRGGB10** = 0x810A0058,
TYPixelFormatMono12 = 0x810C0047, **TYPixelFormatBayerGBRG12** = 0x810C0055, **TYPixelFormat↩BayerBGGR12** = 0x810C0053, **TYPixelFormatBayerGRBG12** = 0x810C0057,
TYPixelFormatBayerRGGB12 = 0x810C0059, **TYPixelFormatMono14** = 0x810E0104, **TYPixelFormat↩BayerGBRG14** = 0x810E0107, **TYPixelFormatBayerBGGR14** = 0x810E0108,
TYPixelFormatBayerGRBG14 = 0x810E0105, **TYPixelFormatBayerRGGB14** = 0x810E0106, **TYPixel↩FormatMono16** = 0x01100007, **TYPixelFormatBayerGBRG16** = 0x01100030,
TYPixelFormatBayerBGGR16 = 0x01100031, **TYPixelFormatBayerGRBG16** = 0x0110002E, **TYPixel↩FormatBayerRGGB16** = 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, **TY↩PixelFormatCoord3D_C16** = 0x011000B8,
TYPixelFormatCoord3D_ABC16 = 0x023000B9, **TYPixelFormatCoord3D_ABC32f** = 0x026000C0, **TY↩PixelFormatJPEG** = 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_PIX↩EL_32BIT** = 0x4 << 28,

TY_PIXEL_10BIT = 0x5 << 28, **TY_PIXEL_12BIT** = 0x6 << 28, **TY_PIXEL_14BIT** = 0x7 << 28, **TY_PI↩
XEL_48BIT** = (uint32_t)0x8 << 28,
TY_PIXEL_64BIT = (uint32_t)0xa << 28 }

- enum TY_PIXEL_FORMAT_LIST : uint32_t {
TY_PIXEL_FORMAT_UNDEFINED = 0, TY_PIXEL_FORMAT_MONO = (TY_PIXEL_8BIT | (0x0 << 24)), TY_PIXEL_FORMAT_BAYER8GB = (TY_PIXEL_8BIT | (0x1 << 24)), TY_PIXEL_FORMAT_BAYER8BG = (TY_PIXEL_8BIT | (0x2 << 24)),
TY_PIXEL_FORMAT_BAYER8GR = (TY_PIXEL_8BIT | (0x3 << 24)), TY_PIXEL_FORMAT_BAYER8RG = (TY_PIXEL_8BIT | (0x4 << 24)), **TY_PIXEL_FORMAT_BAYER8GRBG** = TY_PIXEL_FORMAT_BAY↩
ER8GB, **TY_PIXEL_FORMAT_BAYER8RGGB** = TY_PIXEL_FORMAT_BAYER8BG,
TY_PIXEL_FORMAT_BAYER8GBRG = TY_PIXEL_FORMAT_BAYER8GR, **TY_PIXEL_FORMAT_BAY↩
ER8BGGR** = TY_PIXEL_FORMAT_BAYER8RG, TY_PIXEL_FORMAT_CSI_MONO10 = (TY_PIXEL_10BIT | (0x0 << 24)), TY_PIXEL_FORMAT_CSI_BAYER10GRBG = (TY_PIXEL_10BIT | (0x1 << 24)),
TY_PIXEL_FORMAT_CSI_BAYER10RGGB = (TY_PIXEL_10BIT | (0x2 << 24)), TY_PIXEL_FORMAT_CSI_BAYER10GBRG = (TY_PIXEL_10BIT | (0x3 << 24)), TY_PIXEL_FORMAT_CSI_BAYER10BGGR = (TY_PIXEL_10BIT | (0x4 << 24)), TY_PIXEL_FORMAT_CSI_MONO12 = (TY_PIXEL_12BIT | (0x0 << 24)),
TY_PIXEL_FORMAT_CSI_BAYER12GRBG = (TY_PIXEL_12BIT | (0x1 << 24)), TY_PIXEL_FORMAT_CSI_BAYER12RGGB = (TY_PIXEL_12BIT | (0x2 << 24)), TY_PIXEL_FORMAT_CSI_BAYER12GBRG = (TY_PIXEL_12BIT | (0x3 << 24)), TY_PIXEL_FORMAT_CSI_BAYER12BGGR = (TY_PIXEL_12BIT | (0x4 << 24)),
TY_PIXEL_FORMAT_DEPTH16 = (TY_PIXEL_16BIT | (0x0 << 24)), TY_PIXEL_FORMAT_YVYU = (TY_PIXEL_16BIT | (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_MJPG = (TY_PIXEL_24BIT | (0x3 << 24)), TY_PIXEL_FORMAT_RGB48 = (T↩
Y_PIXEL_48BIT | (0x0 << 24)), TY_PIXEL_FORMAT_BGR48 = (TY_PIXEL_48BIT | (0x1 << 24)),
TY_PIXEL_FORMAT_XYZ48 = (TY_PIXEL_48BIT | (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) }

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 637 of file TYDefs.h.

5.3.3 Typedef Documentation

5.3.3.1 TY_ACC_BIAS

```
typedef struct TY_ACC_BIAS TY_ACC_BIAS
```

a 3x3 matrix

.	.	.
BIASx	BIASy	BIASz

5.3.3.2 TY_ACC_MISALIGNMENT

```
typedef struct TY_ACC_MISALIGNMENT TY_ACC_MISALIGNMENT
```

a 3x3 matrix
|.|.|.|

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

5.3.3.3 TY_ACC_SCALE

`typedef struct TY_ACC_SCALE TY_ACC_SCALE`

a 3x3 matrix

.	.	.
SCALEx	0	0
0	SCALEy	0
0	0	SCALEz

5.3.3.4 TY_ACCESS_MODE_LIST

`typedef enum TY_ACCESS_MODE_LIST TY_ACCESS_MODE_LIST`

Indicate a feature is readable or writable

See also

TYGetFeatureInfo

5.3.3.5 TY_BYTEARRAY_ATTR

`typedef struct TY_BYTEARRAY_ATTR TY_BYTEARRAY_ATTR`

byte array data structure

See also

TYGetByteArray

5.3.3.6 TY_CAMERA_CALIB_INFO

```
typedef struct TY_CAMERA_CALIB_INFO TY_CAMERA_CALIB_INFO
```

camera 's cailbration data

See also

TYGetStruct

5.3.3.7 TY_CAMERA_DISTORTION

```
typedef struct TY_CAMERA_DISTORTION TY_CAMERA_DISTORTION
```

camera distortion parameters

See also

TYGetStruct Usage:

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

5.3.3.8 TY_CAMERA_EXTRINSIC

```
typedef struct TY_CAMERA_EXTRINSIC TY_CAMERA_EXTRINSIC
```

a 4x4 matrix

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

See also

TYGetStruct Usage:

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

5.3.3.9 TY_CAMERA_INTRINSIC

```
typedef struct TY_CAMERA_INTRINSIC TY_CAMERA_INTRINSIC
```

a 3x3 matrix

.	.	.
fx	0	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
```

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

See also

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

5.3.3.15 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 399 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 453 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_ENUM_USER_SET_CURRENT	Set the configuration set to be used as the default startup set. Write & Read
TY_ENUM_USER_SET_SELECTOR	UserSetSelector. Indicates the currently active user configuration profile. Read Only
TY_CMD_USER_SET_LOAD	Selects the configuration set to load, save, or configure. Write & Read
TY_CMD_USER_SET_SAVE	Stores the selected configuration in the camera's volatile memory and sets it as the temporary active configuration. Write Only
TY_STRING_USER_SET_DESCRIPTION	Saves the active configuration set to the chosen user set(set by TY_ENUM_USER_SET_SELECTOR). Write Only
TY_STRUCT_TRIGGER_PARAM	param of trigger, see TY_TRIGGER_PARAM Description of the selected User Set content. Write & Read
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

Enumerator

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
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

Enumerator

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

Enumerator

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 211 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 440 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 545 of file TYDefs.h.

5.3.4.6 TY_PIXEL_FORMAT_LIST

```
enum TY_PIXEL_FORMAT_LIST :  uint32_t
```

pixel format definitions

Enumerator

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

Definition at line 563 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

Enumerator

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

Definition at line 607 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

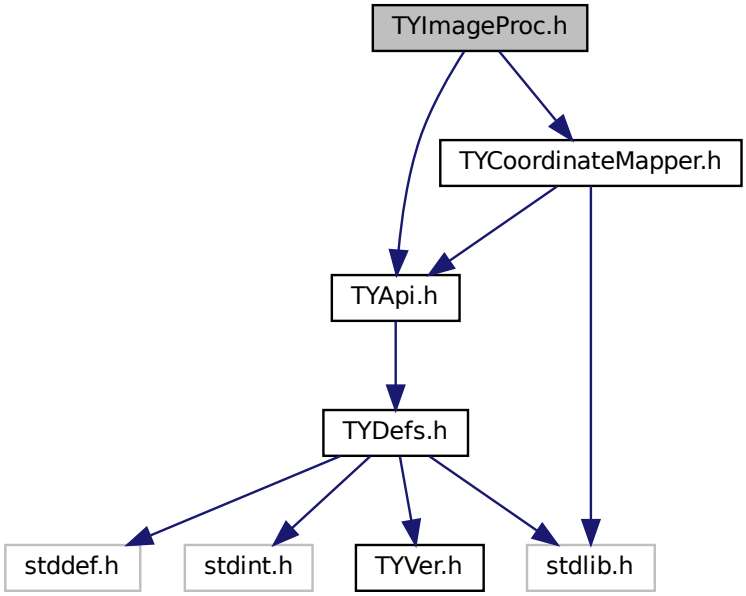
TY_TRIGGER_MODE_OFF	not trigger mode, continuous mode
TY_TRIGGER_MODE_SLAVE	slave mode, receive soft/hardware triggers
TY_TRIGGER_MODE_M_SIG	master mode 1, sending one trigger signal once received a soft trigger
TY_TRIGGER_MODE_M_PER	master mode 2, periodic sending one trigger signals, 'fps' param should 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 708 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)

  Do image undistortion, only support TY_PIXEL_FORMAT_MONO ,TY_PIXEL_FORMAT_RGB,TY_PIXEL_FORM↩ AT_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	*depthImage*	Pointer to depth image array.
in	*imageNum*	Depth image array size.
in,out	*guide*	Guide image.
out	*output*	Output depth image.
in	*param*	Algorithm parameters.

Return values

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

5.4.2.2 TYDepthSpeckleFilter()

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

Remove speckles on depth image.

Parameters

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

Return values

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

5.4.2.3 TYImageProcesAcceEnable()

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

Image processing acceleration switch.

Parameters

in	*en*	Enable image process acceleration switch

5.4.2.4 TYUndistortImage()

```
TY_CAPI TYUndistortImage (
            const TY_CAMERA_CALIB_INFO * srcCalibInfo,
            const TY_IMAGE_DATA * srcImage,
            const TY_CAMERA_INTRINSIC * cameraNewIntrinsic,
            TY_IMAGE_DATA * dstImage )
```

Do image undistortion, only support TY_PIXEL_FORMAT_MONO ,TY_PIXEL_FORMAT_RGB,TY_PIXEL_FOR↩ MAT_BGR.

Parameters

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

Return values

TY_STATUS_OK	Succeed.

Return values

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

Index