

TYCampport3

3

Generated by Doxygen 1.8.14

Contents

1	Main Page	1
1.1	compare to V2:	1
1.2	Note	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	DepthEnhenceParameters Struct Reference	7
4.1.1	Detailed Description	7
4.2	DepthSpeckleFilterParameters Struct Reference	7
4.2.1	Detailed Description	8
4.3	TY_CAMERA_CALIB_INFO Struct Reference	8
4.3.1	Detailed Description	8
4.4	TY_CAMERA_DISTORTION Struct Reference	9
4.4.1	Detailed Description	9
4.5	TY_CAMERA_EXTRINSIC Struct Reference	9
4.5.1	Detailed Description	9
4.6	TY_CAMERA_INTRINSIC Struct Reference	10
4.6.1	Detailed Description	10
4.7	TY_CAMERA_STATISTICS Struct Reference	10
4.7.1	Detailed Description	10

4.8	TY_DEVICE_BASE_INFO Struct Reference	11
4.8.1	Detailed Description	11
4.9	TY_DEVICE_NET_INFO Struct Reference	12
4.9.1	Detailed Description	12
4.10	TY_DEVICE_USB_INFO Struct Reference	12
4.10.1	Detailed Description	12
4.11	TY_ENUM_ENTRY Struct Reference	12
4.11.1	Detailed Description	13
4.12	TY_EVENT_INFO Struct Reference	13
4.12.1	Detailed Description	13
4.13	TY_FEATURE_INFO Struct Reference	13
4.13.1	Detailed Description	14
4.14	TY_FLOAT_RANGE Struct Reference	14
4.14.1	Detailed Description	14
4.15	TY_FRAME_DATA Struct Reference	14
4.15.1	Detailed Description	15
4.16	TY_IMAGE_DATA Struct Reference	15
4.16.1	Detailed Description	16
4.17	TY_INT_RANGE Struct Reference	16
4.17.1	Detailed Description	16
4.18	TY_INTERFACE_INFO Struct Reference	16
4.18.1	Detailed Description	17
4.19	TY_ISP_FEATURE_INFO Struct Reference	17
4.19.1	Detailed Description	17
4.20	TY_PIXEL_DESC Struct Reference	17
4.20.1	Detailed Description	18
4.21	TY_TRIGGER_PARAM Struct Reference	18
4.21.1	Detailed Description	18
4.22	TY_TRIGGER_PARAM_EX Struct Reference	18
4.22.1	Detailed Description	18
4.23	TY_VECT_3F Struct Reference	19
4.23.1	Detailed Description	19
4.24	TY_VERSION_INFO Struct Reference	19
4.24.1	Detailed Description	19

5 File Documentation	21
5.1 TYApi.h File Reference	21
5.1.1 Detailed Description	29
5.1.2 Macro Definition Documentation	29
5.1.2.1 TY_DECLARE_IMAGE_MODE1	29
5.1.3 Typedef Documentation	29
5.1.3.1 TY_CAMERA_CALIB_INFO	29
5.1.3.2 TY_CAMERA_EXTRINSIC	30
5.1.3.3 TY_CAMERA_INTRINSIC	30
5.1.3.4 TY_COMPONENT_ID	30
5.1.3.5 TY_DEVICE_BASE_INFO	30
5.1.3.6 TY_DEVICE_COMPONENT_LIST	31
5.1.3.7 TY_ENUM_ENTRY	31
5.1.3.8 TY_FEATURE_ID	31
5.1.3.9 TY_INTERFACE_INFO	31
5.1.3.10 TY_TRIGGER_ACTIVATION_LIST	32
5.1.3.11 TY_TRIGGER_MODE_LIST	32
5.1.4 Enumeration Type Documentation	32
5.1.4.1 TY_DEVICE_COMPONENT_LIST	32
5.1.4.2 TY_FEATURE_ID_LIST	33
5.1.4.3 TY_PIXEL_FORMAT_LIST	34
5.1.4.4 TY_RESOLUTION_MODE_LIST	34
5.1.4.5 TY_TRIGGER_ACTIVATION_LIST	35
5.1.4.6 TY_TRIGGER_MODE_LIST	35
5.1.5 Function Documentation	35
5.1.5.1 TYClearBufferQueue()	35
5.1.5.2 TYCloseDevice()	36
5.1.5.3 TYCloseInterface()	36
5.1.5.4 TYDeinitLib()	37
5.1.5.5 TYDisableComponents()	37

5.1.5.6	TYEnableComponents()	37
5.1.5.7	TYEnqueueBuffer()	38
5.1.5.8	TYErrorString()	38
5.1.5.9	TYFetchFrame()	39
5.1.5.10	TYForceDeviceIP()	39
5.1.5.11	TYGetBool()	40
5.1.5.12	TYGetByteArray()	40
5.1.5.13	TYGetByteArraySize()	42
5.1.5.14	TYGetComponentIDs()	43
5.1.5.15	TYGetDeviceInfo()	43
5.1.5.16	TYGetDeviceInterface()	43
5.1.5.17	TYGetDeviceList()	44
5.1.5.18	TYGetDeviceNumber()	44
5.1.5.19	TYGetEnabledComponents()	46
5.1.5.20	TYGetEnum()	46
5.1.5.21	TYGetEnumEntryCount()	47
5.1.5.22	TYGetEnumEntryInfo()	47
5.1.5.23	TYGetFeatureInfo()	48
5.1.5.24	TYGetFloat()	49
5.1.5.25	TYGetFloatRange()	49
5.1.5.26	TYGetFrameBufferSize()	50
5.1.5.27	TYGetInt()	50
5.1.5.28	TYGetInterfaceList()	51
5.1.5.29	TYGetInterfaceNumber()	51
5.1.5.30	TYGetIntRange()	52
5.1.5.31	TYGetString()	52
5.1.5.32	TYGetStringLength()	53
5.1.5.33	TYGetStruct()	54
5.1.5.34	TYHasDevice()	54
5.1.5.35	TYHasFeature()	55

5.1.5.36	TYHasInterface()	55
5.1.5.37	TYLibVersion()	56
5.1.5.38	TYOpenDevice()	56
5.1.5.39	TYOpenDeviceWithIP()	57
5.1.5.40	TYOpenInterface()	57
5.1.5.41	TYRegisterEventCallback()	58
5.1.5.42	TYSendSoftTrigger()	58
5.1.5.43	TYSetBool()	59
5.1.5.44	TYSetByteArray()	59
5.1.5.45	TYSetEnum()	60
5.1.5.46	TYSetFloat()	61
5.1.5.47	TYSetInt()	61
5.1.5.48	TYSetString()	62
5.1.5.49	TYSetStruct()	62
5.1.5.50	TYStartCapture()	63
5.1.5.51	TYStopCapture()	64
5.1.5.52	TYUpdateDeviceList()	64
5.1.5.53	TYUpdateInterfaceList()	64
5.2	TYCoordinateMapper.h File Reference	65
5.2.1	Detailed Description	66
5.2.2	Macro Definition Documentation	66
5.2.2.1	TYMAP_CHECKRET	66
5.2.3	Function Documentation	67
5.2.3.1	TYInvertExtrinsic()	67
5.2.3.2	TYMapDepthImageToPoint3d()	67
5.2.3.3	TYMapDepthToPoint3d()	68
5.2.3.4	TYMapPoint3dToDepth()	68
5.2.3.5	TYMapPoint3dToDepthImage()	69
5.2.3.6	TYMapPoint3dToPoint3d()	69
5.3	TYImageProc.h File Reference	70
5.3.1	Detailed Description	71
5.3.2	Function Documentation	71
5.3.2.1	TYDepthEnhanceFilter()	71
5.3.2.2	TYDepthSpeckleFilter()	72
5.3.2.3	TYUndistortImage()	72
5.4	TyIsp.h File Reference	73
5.4.1	Detailed Description	75
5.4.2	Enumeration Type Documentation	75
5.4.2.1	TY_ISP_FEATURE_ID	75

Chapter 1

Main Page

1.1 compare to V2:

1. New Interface Layer Add this layer to specify local network interface to open network camera, solving the problem that someone wants to connect to a network camera with ethernet rather than WIFI. Users have to call interface APIs before opening devices.
2. New Image Processing Library The new library which has header file [TYImageProc.h](#) collects all image processing functions we provided.
3. New Coordinate Mapper New [TYCoordinateMapper.h](#) handles various conversions, including depth <-> point3D, point3D <-> point3D.
4. Components: Removed Point3D component(TY_COMPONENT_POINT3D). Point3D is a virtual component in V2, and the points are calculated from depth image. We put the calculation outside tycam library to increase flexibility.
5. Features: Removed TY_BOOL_TRIGGER_MODE , covered by TY_STRUCT_TRIGGER_PARAM Added TY_STRUCT_CAM_CALIB_DATA , for easy use in image processing library TY_INT_IMAGE_MODE , covered by new added TY_ENUM_IMAGE_MODE Modified TY_ENUM_IMAGE_MODE , means resolution mode in V2, combine resolution and pixel format in V3 Added some network camera's feature, such as TY_INT_PERSISTENT_IP, TY_INT_PERSISTENT_SUBMASK, TY_INT_PACKET_DELAY, etc.

Copyright(C)2016-2019 Percipio All Rights Reserved

1.2 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:

DepthEnhanceParameters	
Default parameter value definition	7
DepthSpeckleFilterParameters	
Default parameter value definition	7
TY_CAMERA_CALIB_INFO	8
TY_CAMERA_DISTORTION	
Camera distortion parameters	9
TY_CAMERA_EXTRINSIC	9
TY_CAMERA_INTRINSIC	10
TY_CAMERA_STATISTICS	10
TY_DEVICE_BASE_INFO	11
TY_DEVICE_NET_INFO	12
TY_DEVICE_USB_INFO	12
TY_ENUM_ENTRY	12
TY_EVENT_INFO	13
TY_FEATURE_INFO	13
TY_FLOAT_RANGE	14
TY_FRAME_DATA	14
TY_IMAGE_DATA	15
TY_INT_RANGE	16
TY_INTERFACE_INFO	16
TY_ISP_FEATURE_INFO	17
TY_PIXEL_DESC	17
TY_TRIGGER_PARAM	18
TY_TRIGGER_PARAM_EX	18
TY_VECT_3F	19
TY_VERSION_INFO	19

Chapter 3

File Index

3.1 File List

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

TYApi.h	TYApi.h includes camera control and data receiving interface, which supports configuration for image resolution, frame rate, exposure time, gain, working mode,etc	21
TYCoordinateMapper.h	Coordinate Conversion API	65
TYImageProc.h	70
Tylsp.h	73

Chapter 4

Class Documentation

4.1 DepthEnhanceParameters Struct Reference

default parameter value definition

```
#include <TYImageProc.h>
```

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

default parameter value definition

Definition at line 50 of file TYImageProc.h.

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

- [TYImageProc.h](#)

4.2 DepthSpeckleFilterParameters Struct Reference

default parameter value definition

```
#include <TYImageProc.h>
```

Public Attributes

- int **max_speckle_size**
- int **max_speckle_diff**

4.2.1 Detailed Description

default parameter value definition

Definition at line 30 of file TYImageProc.h.

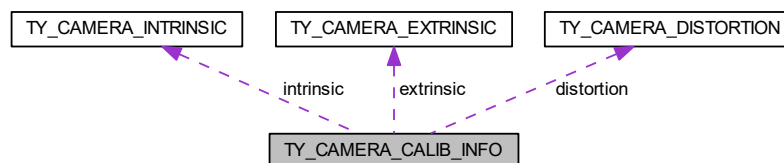
The documentation for this struct was generated from the following file:

- [TYImageProc.h](#)

4.3 TY_CAMERA_CALIB_INFO Struct Reference

```
#include <TYApi.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**

4.3.1 Detailed Description

camera 's caillbration data

See also

[TYGetStruct](#)

Definition at line 549 of file TYApi.h.

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

- [TYApi.h](#)

4.4 TY_CAMERA_DISTORTION Struct Reference

camera distortion parameters

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

Public Attributes

- float [data](#) [12]
Definition is compatible with opencv3.0+ :k1,k2,p1,p2,k3,k4,k5,k6,s1,s2,s3,s4.

4.4.1 Detailed Description

camera distortion parameters

Definition at line 541 of file TYApi.h.

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

- [TYApi.h](#)

4.5 TY_CAMERA_EXTRINSIC Struct Reference

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

Public Attributes

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

4.5.1 Detailed Description

a 4x4 matrix

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

Definition at line 535 of file TYApi.h.

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

- [TYApi.h](#)

4.6 TY_CAMERA_INTRINSIC Struct Reference

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

Public Attributes

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

4.6.1 Detailed Description

a 3x3 matrix

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

Definition at line 523 of file TYApi.h.

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

- [TYApi.h](#)

4.7 TY_CAMERA_STATISTICS Struct Reference

Public Attributes

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

4.7.1 Detailed Description

Definition at line 580 of file TYApi.h.

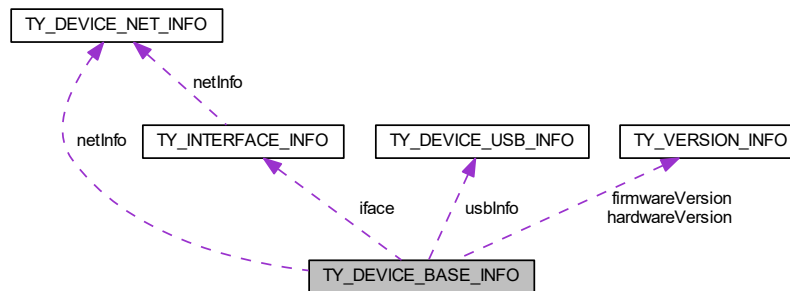
The documentation for this struct was generated from the following file:

- [TYApi.h](#)

4.8 TY_DEVICE_BASE_INFO Struct Reference

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

Collaboration diagram for TY_DEVICE_BASE_INFO:



Public Attributes

- [TY_INTERFACE_INFO](#) **iface**
- char **id** [32]
device serial number
- char **vendorName** [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 **reserved** [256]

4.8.1 Detailed Description

See also

[TYGetDeviceList](#)

Definition at line 456 of file TYApi.h.

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

- [TYApi.h](#)

4.9 TY_DEVICE_NET_INFO Struct Reference

Public Attributes

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

4.9.1 Detailed Description

Definition at line 428 of file TYApi.h.

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

- [TYApi.h](#)

4.10 TY_DEVICE_USB_INFO Struct Reference

Public Attributes

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

4.10.1 Detailed Description

Definition at line 438 of file TYApi.h.

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

- [TYApi.h](#)

4.11 TY_ENUM_ENTRY Struct Reference

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

Public Attributes

- char **description** [64]
- int32_t **value**
- int32_t **reserved** [3]

4.11.1 Detailed Description

enum feature entry information

See also

[TYGetEnumEntryInfo](#)

Definition at line 503 of file TYApi.h.

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

- [TYApi.h](#)

4.12 TY_EVENT_INFO Struct Reference

Public Attributes

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

4.12.1 Detailed Description

Definition at line 618 of file TYApi.h.

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

- [TYApi.h](#)

4.13 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
- int32_t [bindComponentID](#)
component ID current feature bind to
- int32_t [bindFeatureID](#)
feature ID current feature bind to
- char **reserved** [252]

4.13.1 Detailed Description

Definition at line 471 of file TYApi.h.

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

- [TYApi.h](#)

4.14 TY_FLOAT_RANGE Struct Reference

Public Attributes

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

4.14.1 Detailed Description

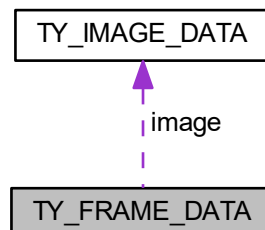
Definition at line 493 of file TYApi.h.

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

- [TYApi.h](#)

4.15 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.
- `TY_IMAGE_DATA image [10]`
Buffer data, max to 10 images per frame, each buffer data could be an image or something else.

4.15.1 Detailed Description

Definition at line 608 of file TYApi.h.

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

- [TYApi.h](#)

4.16 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.
- `int32_t 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.
- `int32_t pixelFormat`
Pixel format, see TY_PIXEL_FORMAT_LIST.
- `int32_t reserved [9]`
Reserved.

4.16.1 Detailed Description

Definition at line 593 of file TYApi.h.

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

- [TYApi.h](#)

4.17 TY_INT_RANGE Struct Reference

Public Attributes

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

4.17.1 Detailed Description

Definition at line 485 of file TYApi.h.

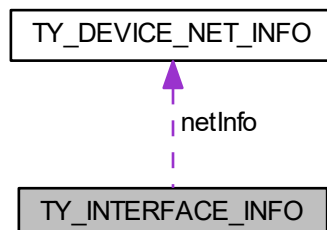
The documentation for this struct was generated from the following file:

- [TYApi.h](#)

4.18 TY_INTERFACE_INFO Struct Reference

```
#include <TYApi.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**

4.18.1 Detailed Description

See also

[TYGetInterfaceList](#)

Definition at line 446 of file TYApi.h.

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

- [TYApi.h](#)

4.19 TY_ISP_FEATURE_INFO Struct Reference

Public Attributes

- [TY_ISP_FEATURE_ID](#) **id**
- int32_t **size**
- const char * **name**
- const char * **value_type**
- TY_ACCESS_MODE **mode**

4.19.1 Detailed Description

Definition at line 63 of file TyIsp.h.

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

- [TyIsp.h](#)

4.20 TY_PIXEL_DESC Struct Reference

Public Attributes

- int16_t **x**
- int16_t **y**
- uint16_t **depth**
- uint16_t **rsvd**

4.20.1 Detailed Description

Definition at line 12 of file TYCoordinateMapper.h.

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

- [TYCoordinateMapper.h](#)

4.21 TY_TRIGGER_PARAM Struct Reference

Public Attributes

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

4.21.1 Detailed Description

Definition at line 560 of file TYApi.h.

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

- [TYApi.h](#)

4.22 TY_TRIGGER_PARAM_EX Struct Reference

Public Attributes

- TY_TRIGGER_MODE **mode**
- int8_t **fps**
- int8_t **duty**
- int32_t **laser_stream**
- int32_t **led_stream**
- int32_t **led_expo**
- int32_t **led_gain**
- int32_t **rsvd** [20]

4.22.1 Detailed Description

Definition at line 568 of file TYApi.h.

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

- [TYApi.h](#)

4.23 TY_VECT_3F Struct Reference

Public Attributes

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

4.23.1 Detailed Description

Definition at line 510 of file TYApi.h.

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

- [TYApi.h](#)

4.24 TY_VERSION_INFO Struct Reference

Public Attributes

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

4.24.1 Detailed Description

Definition at line 420 of file TYApi.h.

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

- [TYApi.h](#)

Chapter 5

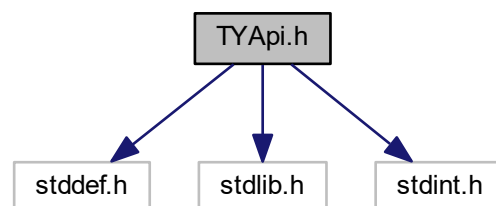
File Documentation

5.1 TYApi.h File Reference

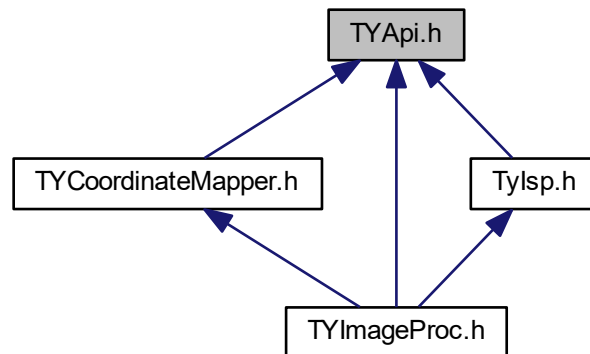
[TYApi.h](#) includes camera control and data receiving interface, which supports configuration for image resolution, frame rate, exposure time, gain, working mode, etc.

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

Include dependency graph for TYApi.h:



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



Classes

- struct [TY_VERSION_INFO](#)
- struct [TY_DEVICE_NET_INFO](#)
- 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](#)
- struct [TY_ENUM_ENTRY](#)
- struct [TY_VECT_3F](#)
- struct [TY_CAMERA_INTRINSIC](#)
- struct [TY_CAMERA_EXTRINSIC](#)
- struct [TY_CAMERA_DISTORTION](#)

camera distortion parameters

- struct [TY_CAMERA_CALIB_INFO](#)
- struct [TY_TRIGGER_PARAM](#)
- struct [TY_TRIGGER_PARAM_EX](#)
- struct [TY_CAMERA_STATISTICS](#)
- struct [TY_IMAGE_DATA](#)
- struct [TY_FRAME_DATA](#)
- struct [TY_EVENT_INFO](#)

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_LIB_VERSION_MAJOR 3`
- `#define TY_LIB_VERSION_MINOR 4`
- `#define TY_LIB_VERSION_PATCH 1`
- `#define TY_DECLARE_IMAGE_MODE0(pix, res) TY_IMAGE_MODE_##pix##_##res = TY_PIXEL_FORMAT_##pix | TY_RESOLUTION_MODE_##res`
- `#define TY_DECLARE_IMAGE_MODE1(pix)`
- `#define TY_CAPI TY_EXTC TY_EXPORT TY_STATUS TY_STDC`

Typedefs

- `typedef enum TY_STATUS_LIST TY_STATUS_LIST`
API call return status.
- `typedef int32_t TY_STATUS`
- `typedef enum TY_EVENT_LIST TY_EVENT_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 int32_t TY_COMPONENT_ID`
component unique id
- `typedef enum TY_FEATURE_TYPE_LIST TY_FEATURE_TYPE_LIST`
Feature Format Type definitions.
- `typedef int32_t TY_FEATURE_TYPE`
- `typedef enum TY_FEATURE_ID_LIST TY_FEATURE_ID_LIST`
feature for component definitions
- `typedef int32_t TY_FEATURE_ID`
feature unique id
- `typedef enum TY_TRIGGER_ACTIVATION_LIST TY_TRIGGER_ACTIVATION_LIST`
set external trigger signal edge
- `typedef int32_t TY_TRIGGER_ACTIVATION`
- `typedef enum TY_INTERFACE_TYPE_LIST TY_INTERFACE_TYPE_LIST`
interface type definition
- `typedef int32_t TY_INTERFACE_TYPE`
- `typedef enum TY_ACCESS_MODE_LIST TY_ACCESS_MODE_LIST`
a feature is readable or writable
- `typedef int8_t TY_ACCESS_MODE`
- `typedef enum TY_PIXEL_BITS_LIST TY_PIXEL_BITS_LIST`
Pixel size type definitions.
- `typedef enum TY_PIXEL_FORMAT_LIST TY_PIXEL_FORMAT_LIST`
pixel format definitions
- `typedef int32_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 int32_t **TY_IMAGE_MODE**
- typedef enum **TY_TRIGGER_MODE_LIST** **TY_TRIGGER_MODE_LIST**
- typedef int16_t **TY_TRIGGER_MODE**
- typedef struct **TY_VERSION_INFO** **TY_VERSION_INFO**
- typedef struct **TY_DEVICE_NET_INFO** **TY_DEVICE_NET_INFO**
- 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 struct **TY_FEATURE_INFO** **TY_FEATURE_INFO**
- typedef struct **TY_INT_RANGE** **TY_INT_RANGE**
- typedef struct **TY_FLOAT_RANGE** **TY_FLOAT_RANGE**
- 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**
- camera distortion parameters*
- 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_CAMERA_STATISTICS** **TY_CAMERA_STATISTICS**
- 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 void(* **TY_EVENT_CALLBACK**) (**TY_EVENT_INFO** *, void *userdata)

Enumerations

- enum **TY_STATUS_LIST** {
TY_STATUS_OK = 0, **TY_STATUS_ERROR** = -1001, **TY_STATUS_NOT_INITED** = -1002, **TY_STATUS_**↵
NOT_IMPLEMENTED = -1003,
TY_STATUS_NOT_PERMITTED = -1004, **TY_STATUS_DEVICE_ERROR** = -1005, **TY_STATUS_INVA**↵
LID_PARAMETER = -1006, **TY_STATUS_INVALID_HANDLE** = -1007,
TY_STATUS_INVALID_COMPONENT = -1008, **TY_STATUS_INVALID_FEATURE** = -1009, **TY_STATU**↵
S_WRONG_TYPE = -1010, **TY_STATUS_WRONG_SIZE** = -1011,
TY_STATUS_OUT_OF_MEMORY = -1012, **TY_STATUS_OUT_OF_RANGE** = -1013, **TY_STATUS_TIM**↵
EOUT = -1014, **TY_STATUS_WRONG_MODE** = -1015,
TY_STATUS_BUSY = -1016, **TY_STATUS_IDLE** = -1017, **TY_STATUS_NO_DATA** = -1018, **TY_STATU**↵
S_NO_BUFFER = -1019,
TY_STATUS_NULL_POINTER = -1020, **TY_STATUS_READONLY_FEATURE** = -1021, **TY_STATUS_I**↵
NVALID_DESCRIPTOR = -1022, **TY_STATUS_INVALID_INTERFACE** = -1023,
TY_STATUS_FIRMWARE_ERROR = -1024 }
 - API call return status.*
- enum **TY_EVENT_LIST** { **TY_EVENT_DEVICE_OFFLINE** = -2001, **TY_EVENT_LICENSE_ERROR** = -
2002, **TY_EVENT_FW_INIT_ERROR** = -2003 }
- enum **TY_DEVICE_COMPONENT_LIST** {
TY_COMPONENT_DEVICE = 0x80000000, **TY_COMPONENT_DEPTH_CAM** = 0x00010000, **TY_COMPONENT_IR_CAM_L**↵
= 0x00040000, **TY_COMPONENT_IR_CAM_RIGHT** = 0x00080000,
TY_COMPONENT_RGB_CAM_LEFT = 0x00100000, **TY_COMPONENT_RGB_CAM_RIGHT** = 0x00200000,
TY_COMPONENT_LASER = 0x00400000, **TY_COMPONENT_IMU** = 0x00800000,
TY_COMPONENT_BRIGHT_HISTO = 0x01000000, **TY_COMPONENT_STORAGE** = 0x02000000,
TY_COMPONENT_RGB_CAM = **TY_COMPONENT_RGB_CAM_LEFT** }

- enum `TY_FEATURE_TYPE_LIST` {
`TY_FEATURE_INT` = 0x1000, `TY_FEATURE_FLOAT` = 0x2000, `TY_FEATURE_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` {
`TY_STRUCT_CAM_INTRINSIC` = 0x0000 | `TY_FEATURE_STRUCT`, `TY_STRUCT_EXTRINSIC_TO_LEFT_IR` = 0x0001 | `TY_FEATURE_STRUCT`, `TY_STRUCT_CAM_DISTORTION` = 0x0006 | `TY_FEATURE_STRUCT`,
`TY_STRUCT_CAM_CALIB_DATA` = 0x0007 | `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_INT_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_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_ACTIVATION` = 0x0201 | `TY_FEATURE_ENUM`,
`TY_INT_FRAME_PER_TRIGGER` = 0x0202 | `TY_FEATURE_INT`,
`TY_STRUCT_TRIGGER_PARAM` = 0x0523 | `TY_FEATURE_STRUCT`, `TY_STRUCT_TRIGGER_PARAM_EX` = 0x0525 | `TY_FEATURE_STRUCT`, `TY_BOOL_KEEP_ALIVE_ONOFF` = 0x0203 | `TY_FEATURE_BOOL`,
`TY_INT_KEEP_ALIVE_TIMEOUT` = 0x0204 | `TY_FEATURE_INT`,
`TY_BOOL_CMOS_SYNC` = 0x0205 | `TY_FEATURE_BOOL`, `TY_INT_TRIGGER_DELAY_US` = 0x0206 | `TY_FEATURE_INT`, `TY_BOOL_TRIGGER_OUT_IO` = 0x0207 | `TY_FEATURE_BOOL`, `TY_INT_TRIGGER_DURATION_US` = 0x0208 | `TY_FEATURE_INT`,
`TY_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_INT_LASER_POWER` = 0x0500 | `TY_FEATURE_INT`, `TY_BOOL_LASER_AUTO_CTRL` = 0x0501 | `TY_FEATURE_BOOL`, `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` }
feature for component definitions
- enum `TY_TRIGGER_ACTIVATION_LIST` { `TY_TRIGGER_ACTIVATION_FALLINGEDGE` = 0, `TY_TRIGGER_ACTIVATION_RISINGEDGE` = 1 }
set external trigger signal edge
- enum `TY_INTERFACE_TYPE_LIST` {
`TY_INTERFACE_UNKNOWN` = 0, `TY_INTERFACE_RAW` = 1, `TY_INTERFACE_USB` = 2, `TY_INTERFACE_ETHERNET` = 4,
`TY_INTERFACE_IEEE80211` = 8, `TY_INTERFACE_ALL` = 0xffff }
interface type definition
- enum `TY_ACCESS_MODE_LIST` { `TY_ACCESS_READABLE` = 0x1, `TY_ACCESS_WRITABLE` = 0x2 }
a feature is readable or writable
- enum `TY_PIXEL_BITS_LIST` { `TY_PIXEL_8BIT` = 0x1 << 28, `TY_PIXEL_16BIT` = 0x2 << 28, `TY_PIXEL_24BIT` = 0x3 << 28, `TY_PIXEL_32BIT` = 0x4 << 28 }
Pixel size type definitions.
- enum `TY_PIXEL_FORMAT_LIST` {
`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_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_RGB = (TY_PIXEL_24BIT | (0x0 << 24)),
TY_PIXEL_FORMAT_BGR = (TY_PIXEL_24BIT | (0x1 << 24)),
TY_PIXEL_FORMAT_JPEG = (TY_PIXEL_24BIT | (0x2 << 24)), TY_PIXEL_FORMAT_MJPEG = (TY_PIXEL_24BIT | (0x3 << 24)) }
```

pixel format definitions

- enum **TY_RESOLUTION_MODE_LIST** {
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_2592x1944** = (2592<<12)+1944 }

predefined resolution list

- enum **TY_IMAGE_MODE_LIST** {
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** {
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 }

Functions

- TY_EXTC TY_EXPORT const char *TY_STDC **TYErrorString** (TY_STATUS errorID)
Get error information.
- TY_CAPI **TYDeinitLib** (void)
Deinit this library.
- TY_CAPI **TYLibVersion** (TY_VERSION_INFO *version)
Get current library version.
- TY_CAPI **TYUpdateInterfaceList** ()
Update current interfaces. call before TYGetInterfaceList.
- TY_CAPI **TYGetInterfaceNumber** (uint32_t *pNumIfaces)
Get number of current interfaces.
- TY_CAPI **TYGetInterfaceList** (TY_INTERFACE_INFO *pIfaceInfos, uint32_t bufferCount, uint32_t *filledCount)
Get interface info list.
- TY_CAPI **TYHasInterface** (const char *ifaceID, bool *value)
Check if has interface.
- TY_CAPI **TYOpenInterface** (const char *ifaceID, TY_INTERFACE_HANDLE *outHandle)
Open specified interface.
- TY_CAPI **TYCloseInterface** (TY_INTERFACE_HANDLE ifaceHandle)
Close interface.
- TY_CAPI **TYUpdateDeviceList** (TY_INTERFACE_HANDLE ifaceHandle)
Update current connected devices.
- TY_CAPI **TYGetDeviceNumber** (TY_INTERFACE_HANDLE ifaceHandle, uint32_t *deviceNumber)

- Get number of current connected devices.*
- TY_CAPI [TYGetDeviceList](#) ([TY_INTERFACE_HANDLE](#) ifaceHandle, [TY_DEVICE_BASE_INFO](#) *deviceInfos, [uint32_t](#) bufferCount, [uint32_t](#) *filledDeviceCount)
- Get device info list.*
- TY_CAPI [TYHasDevice](#) ([TY_INTERFACE_HANDLE](#) ifaceHandle, [const char](#) *deviceId, [bool](#) *value)
- Check whether the interface has the specified device.*
- TY_CAPI [TYOpenDevice](#) ([TY_INTERFACE_HANDLE](#) ifaceHandle, [const char](#) *deviceId, [TY_DEV_HANDLE](#) *outDeviceHandle)
- 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)
- 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, [int32_t](#) *componentIDs)
- Get all components IDs.*
- TY_CAPI [TYGetEnabledComponents](#) ([TY_DEV_HANDLE](#) hDevice, [int32_t](#) *componentIDs)
- Get all enabled components IDs.*
- TY_CAPI [TYEnableComponents](#) ([TY_DEV_HANDLE](#) hDevice, [int32_t](#) componentIDs)
- Enable components.*
- TY_CAPI [TYDisableComponents](#) ([TY_DEV_HANDLE](#) hDevice, [int32_t](#) 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 [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, int32_t *value)
Get current value of enum feature.
- TY_CAPI [TYSetEnum](#) (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, int32_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 [_TYInitLib](#) (void)

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 Macro Definition Documentation

5.1.2.1 TY_DECLARE_IMAGE_MODE1

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

Value:

```
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, 2592x1944)
```

Definition at line 372 of file TYApi.h.

5.1.3 Typedef Documentation

5.1.3.1 TY_CAMERA_CALIB_INFO

```
typedef struct TY_CAMERA_CALIB_INFO TY_CAMERA_CALIB_INFO
```

camera 's caillbration data

See also

[TYGetStruct](#)

5.1.3.2 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

5.1.3.3 TY_CAMERA_INTRINSIC

```
typedef struct TY_CAMERA_INTRINSIC TY_CAMERA_INTRINSIC
```

a 3x3 matrix

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

5.1.3.4 TY_COMPONENT_ID

```
typedef int32_t TY_COMPONENT_ID
```

component unique id

See also

[TY_DEVICE_COMPONENT_LIST](#)

Definition at line 209 of file TYApi.h.

5.1.3.5 TY_DEVICE_BASE_INFO

```
typedef struct TY_DEVICE_BASE_INFO TY_DEVICE_BASE_INFO
```

See also

[TYGetDeviceList](#)

5.1.3.6 TY_DEVICE_COMPONENT_LIST

```
typedef enum TY_DEVICE_COMPONENT_LIST TY_DEVICE_COMPONENT_LIST
```

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

5.1.3.7 TY_ENUM_ENTRY

```
typedef struct TY_ENUM_ENTRY TY_ENUM_ENTRY
```

enum feature entry information

See also

[TYGetEnumEntryInfo](#)

5.1.3.8 TY_FEATURE_ID

```
typedef int32_t TY_FEATURE_ID
```

feature unique id

See also

[TY_FEATURE_ID_LIST](#)

Definition at line 289 of file TYApi.h.

5.1.3.9 TY_INTERFACE_INFO

```
typedef struct TY_INTERFACE_INFO TY_INTERFACE_INFO
```

See also

[TYGetInterfaceList](#)

5.1.3.10 TY_TRIGGER_ACTIVATION_LIST

```
typedef enum TY_TRIGGER_ACTIVATION_LIST TY_TRIGGER_ACTIVATION_LIST
```

set external trigger signal edge

See also

refer to sample SimpleView_TriggerMode for detail usage

5.1.3.11 TY_TRIGGER_MODE_LIST

```
typedef enum TY_TRIGGER_MODE_LIST TY_TRIGGER_MODE_LIST
```

See also

refer to sample SimpleView_TriggerMode for detail usage

5.1.4 Enumeration Type Documentation

5.1.4.1 TY_DEVICE_COMPONENT_LIST

```
enum TY_DEVICE_COMPONENT_LIST
```

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 194 of file TYApi.h.

5.1.4.2 TY_FEATURE_ID_LIST

enum [TY_FEATURE_ID_LIST](#)

feature for component definitions

Enumerator

TY_STRUCT_CAM_INTRINSIC	see TY_CAMERA_INTRINSIC
TY_STRUCT_EXTRINSIC_TO_LEFT_IR	extrinsic from current component to left IR, see TY_CAMERA_EXTRINSIC
TY_STRUCT_CAM_DISTORTION	see TY_CAMERA_DISTORTION
TY_STRUCT_CAM_CALIB_DATA	see TY_CAMERA_CALIB_INFO
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_STRUCT_CAM_STATISTICS	statistical information, see TY_CAMERA_STATISTICS
TY_INT_WIDTH	Image width.
TY_INT_HEIGHT	Image height.
TY_ENUM_IMAGE_MODE	Resolution-PixelFormat mode, see TY_IMAGE_MODE_LIST .
TY_ENUM_TRIGGER_ACTIVATION	Trigger activation, see TY_TRIGGER_ACTIVATION_LIST .
TY_INT_FRAME_PER_TRIGGER	Number of frames captured per trigger.
TY_STRUCT_TRIGGER_PARAM	param of trigger, see TY_TRIGGER_PARAM
TY_STRUCT_TRIGGER_PARAM_EX	param of trigger, see TY_TRIGGER_PARAM_EX
TY_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_BOOL_AUTO_EXPOSURE	Auto exposure switch.
TY_INT_EXPOSURE_TIME	Exposure time in percentage.
TY_BOOL_AUTO_GAIN	Auto gain switch.
TY_INT_GAIN	Sensor Gain.
TY_BOOL_AUTO_AWB	Auto white balance.
TY_INT_LASER_POWER	Laser power level.
TY_BOOL_LASER_AUTO_CTRL	Laser auto ctrl.
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.

Definition at line 228 of file TYApi.h.

5.1.4.3 TY_PIXEL_FORMAT_LIST

enum [TY_PIXEL_FORMAT_LIST](#)

pixel format definitions

Enumerator

TY_PIXEL_FORMAT_MONO	0x10000000
TY_PIXEL_FORMAT_BAYER8GB	0x11000000
TY_PIXEL_FORMAT_DEPTH16	0x20000000
TY_PIXEL_FORMAT_YVYU	0x21000000, yvyu422
TY_PIXEL_FORMAT_YUYV	0x22000000, yuyv422
TY_PIXEL_FORMAT_RGB	0x30000000
TY_PIXEL_FORMAT_BGR	0x31000000
TY_PIXEL_FORMAT_JPEG	0x32000000
TY_PIXEL_FORMAT_MJPEG	0x33000000

Definition at line 334 of file TYApi.h.

5.1.4.4 TY_RESOLUTION_MODE_LIST

enum [TY_RESOLUTION_MODE_LIST](#)

predefined resolution list

Enumerator

TY_RESOLUTION_MODE_160x120	0x000a0078
TY_RESOLUTION_MODE_240x320	0x000f0140
TY_RESOLUTION_MODE_320x180	0x001400b4
TY_RESOLUTION_MODE_320x200	0x001400c8
TY_RESOLUTION_MODE_320x240	0x001400f0
TY_RESOLUTION_MODE_480x640	0x001e0280
TY_RESOLUTION_MODE_640x360	0x00280168
TY_RESOLUTION_MODE_640x400	0x00280190
TY_RESOLUTION_MODE_640x480	0x002801e0
TY_RESOLUTION_MODE_960x1280	0x003c0500
TY_RESOLUTION_MODE_1280x720	0x005002d0
TY_RESOLUTION_MODE_1280x800	0x00500320
TY_RESOLUTION_MODE_1280x960	0x005003c0
TY_RESOLUTION_MODE_2592x1944	0x00a20798

Definition at line 350 of file TYApi.h.

5.1.4.5 TY_TRIGGER_ACTIVATION_LIST

enum [TY_TRIGGER_ACTIVATION_LIST](#)

set external trigger signal edge

See also

refer to sample SimpleView_TriggerMode for detail usage

Definition at line 294 of file TYApi.h.

5.1.4.6 TY_TRIGGER_MODE_LIST

enum [TY_TRIGGER_MODE_LIST](#)

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/hardware trigger
TY_TRIGGER_MODE_M_PER	master mode 2, periodic sending one trigger signals, 'fps' param should be set

Definition at line 406 of file TYApi.h.

5.1.5 Function Documentation

5.1.5.1 TYClearBufferQueue()

```
TY_CAPI TYClearBufferQueue (
    TY\_DEV\_HANDLE hDevice )
```

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

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_BUSY</i>	Device is capturing.

5.1.5.2 TYCloseDevice()

```
TY_CAPI TYCloseDevice (
    TY_DEV_HANDLE hDevice )
```

Close device by device handle.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_IDLE</i>	Device has been closed.

5.1.5.3 TYCloseInterface()

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

Close interface.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	Interface not found.

5.1.5.4 TYDeinitLib()

```
TY_CAPI TYDeinitLib (
    void )
```

Deinit this library.

Return values

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

5.1.5.5 TYDisableComponents()

```
TY_CAPI TYDisableComponents (
    TY_DEV_HANDLE hDevice,
    int32_t componentIDs )
```

Disable components.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Some components specified by componentIDs are invalid.
<i>TY_STATUS_BUSY</i>	Device is capturing.

See also

[TY_DEVICE_COMPONENT_LIST](#)

5.1.5.6 TYEnableComponents()

```
TY_CAPI TYEnableComponents (
    TY_DEV_HANDLE hDevice,
    int32_t componentIDs )
```

Enable components.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Some components specified by componentIDs are invalid.
<i>TY_STATUS_BUSY</i>	Device is capturing.

5.1.5.7 TYEnqueueBuffer()

```

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

```

Enqueue a user allocated buffer.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_NULL_POINTER</i>	buffer is NULL.
<i>TY_STATUS_WRONG_SIZE</i>	The input buffer is not large enough.

5.1.5.8 TYErrorString()

```

TY_EXTC TY_EXPORT const char *TY_STDC TYErrorString (
    TY_STATUS errorID )

```

Get error information.

Parameters

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

Returns

Error string.

5.1.5.9 TYFetchFrame()

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

Fetch one frame.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_NULL_POINTER</i>	frame is NULL.
<i>TY_STATUS_IDLE</i>	Device capturing is not started.
<i>TY_STATUS_WRONG_MODE</i>	Callback has been registered, this function is disabled.
<i>TY_STATUS_TIMEOUT</i>	Timeout.

5.1.5.10 TYForceDeviceIP()

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

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

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	Invalid interface handle.
<i>TY_STATUS_WRONG_TYPE</i>	Wrong interface type, should be network.
<i>TY_STATUS_NULL_POINTER</i>	MAC or newIP/newNetMask/newGateway is NULL.
<i>TY_STATUS_INVALID_PARAMETER</i>	MAC is not valid.
<i>TY_STATUS_TIMEOUT</i>	No device found.
<i>TY_STATUS_DEVICE_ERROR</i>	Set new IP failed.

5.1.5.11 TYGetBool()

```

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

```

Get value of bool feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_BOOL.
<i>TY_STATUS_NULL_POINTER</i>	value is NULL.

5.1.5.12 TYGetByteArray()

```

TY_CAPI TYGetByteArray (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    uint8_t * pBuffer,
    uint32_t bufferSize )

```


Read byte array from device.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_BYTEARRAY.
<i>TY_STATUS_NULL_POINTER</i>	pbuffer is NULL.
<i>TY_STATUS_WRONG_SIZE</i>	bufferSize incorrect.

5.1.5.13 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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
out	<i>pSize</i>	size of specified byte array zone.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_BYTEARRAY.
<i>TY_STATUS_NULL_POINTER</i>	pSize is NULL.

5.1.5.14 TYGetComponentIDs()

```
TY_CAPI TYGetComponentIDs (
    TY_DEV_HANDLE hDevice,
    int32_t * componentIDs )
```

Get all components IDs.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_NULL_POINTER</i>	componentIDs is NULL.

See also

[TY_DEVICE_COMPONENT_LIST](#)

5.1.5.15 TYGetDeviceInfo()

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

Get base info of the open device.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_NULL_POINTER</i>	componentIDs is NULL.

5.1.5.16 TYGetDeviceInterface()

```
TY_CAPI TYGetDeviceInterface (
```

```

    TY_DEV_HANDLE hDevice,
    TY_INTERFACE_HANDLE * pIface )

```

Get interface handle by device handle.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_NULL_POINTER</i>	plface is NULL.

5.1.5.17 TYGetDeviceList()

```

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

```

Get device info list.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	Invalid interface handle.
<i>TY_STATUS_NULL_POINTER</i>	deviceInfos or filledDeviceCount is NULL.

5.1.5.18 TYGetDeviceNumber()

```

TY_CAPI TYGetDeviceNumber (
    TY_INTERFACE_HANDLE ifaceHandle,
    uint32_t * deviceNumber )

```

Get number of current connected devices.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	Invalid interface handle.
<i>TY_STATUS_NULL_POINTER</i>	deviceNumber is NULL.

5.1.5.19 TYGetEnabledComponents()

```
TY_CAPI TYGetEnabledComponents (
    TY_DEV_HANDLE hDevice,
    int32_t * componentIDs )
```

Get all enabled components IDs.

Parameters

in	<i>hDevice</i>	Device handle.
out	<i>componentIDs</i>	Enabled component IDs.(bit flag)

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_NULL_POINTER</i>	componentIDs is NULL.

See also

[TY_DEVICE_COMPONENT_LIST](#)

5.1.5.20 TYGetEnum()

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

Get current value of enum feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_ENUM.
<i>TY_STATUS_NULL_POINTER</i>	value is NULL.

5.1.5.21 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	<i>hDevice</i>	Device handle.
in	<i>componentID</i>	Component ID.
in	<i>featureID</i>	Feature ID.
out	<i>entryCount</i>	Entry count.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_ENUM.
<i>TY_STATUS_NULL_POINTER</i>	entryCount is NULL.

5.1.5.22 TYGetEnumEntryInfo()

```

TY_CAPI TYGetEnumEntryInfo (

```

```

    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    TY_ENUM_ENTRY * entries,
    uint32_t entryCount,
    uint32_t * filledEntryCount )

```

Get list of enum entries.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_ENUM.
<i>TY_STATUS_NULL_POINTER</i>	entries or filledEntryCount is NULL.

5.1.5.23 TYGetFeatureInfo()

```

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

```

Get feature info.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.

Return values

<i>TY_STATUS_NULL_POINTER</i>	featureInfo is NULL.
-------------------------------	----------------------

5.1.5.24 TYGetFloat()

```

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

```

Get value of float feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_FLOAT.
<i>TY_STATUS_NULL_POINTER</i>	value is NULL.

5.1.5.25 TYGetFloatRange()

```

TY_CAPI TYGetFloatRange (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    TY_FLOAT_RANGE * floatRange )

```

Get value range of float feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_FLOAT.
<i>TY_STATUS_NULL_POINTER</i>	floatRange is NULL.

5.1.5.26 TYGetFrameBufferSize()

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

Get total buffer size of one frame in current configuration.

Parameters

in	<i>hDevice</i>	Device handle.
out	<i>bufferSize</i>	Buffer size per frame.

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_NULL_POINTER</i>	bufferSize is NULL.

5.1.5.27 TYGetInt()

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

Get value of integer feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_INT.
<i>TY_STATUS_NULL_POINTER</i>	value is NULL.

5.1.5.28 TYGetInterfaceList()

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

Get interface info list.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_NULL_POINTER</i>	pIfaceInfos or filledCount is NULL.

5.1.5.29 TYGetInterfaceNumber()

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

Get number of current interfaces.

Parameters

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

Return values

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

Return values

<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_NULL_POINTER</i>	deviceNumber is NULL.

5.1.5.30 TYGetIntRange()

```

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

```

Get value range of integer feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_INT.
<i>TY_STATUS_NULL_POINTER</i>	intRange is NULL.

5.1.5.31 TYGetString()

```

TY_CAPI TYGetString (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    char * buffer,
    uint32_t bufferSize )

```

Get value of string feature.

Parameters

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

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_STRING.
<i>TY_STATUS_NULL_POINTER</i>	buffer is NULL.

See also

[TYGetStringLength](#)

5.1.5.32 TYGetStringLength()

```

TY_CAPI TYGetStringLength (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    uint32_t * size )

```

Get internal buffer size of string feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_STRING.
<i>TY_STATUS_NULL_POINTER</i>	size is NULL.

See also

[TYGetString](#)

5.1.5.33 TYGetStruct()

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

Get value of struct.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_STRUCT.
<i>TY_STATUS_NULL_POINTER</i>	pStruct is NULL.
<i>TY_STATUS_WRONG_SIZE</i>	structSize incorrect.

5.1.5.34 TYHasDevice()

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

Check whether the interface has the specified device.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	Invalid interface handle.
<i>TY_STATUS_NULL_POINTER</i>	deviceId or value is NULL.

5.1.5.35 TYHasFeature()

```

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

```

Check whether a component has a specific feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_NULL_POINTER</i>	value is NULL.

5.1.5.36 TYHasInterface()

```

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

```

Check if has interface.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_NULL_POINTER</i>	ifaceID or outHandle is NULL.

See also

[TYGetInterfaceList](#)

5.1.5.37 TYLibVersion()

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

Get current library version.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NULL_POINTER</i>	buffer is NULL.

5.1.5.38 TYOpenDevice()

```
TY_CAPI TYOpenDevice (
    TY_INTERFACE_HANDLE ifaceHandle,
    const char * deviceID,
    TY_DEV_HANDLE * outDeviceHandle )
```

Open device by device ID.

Parameters

in	<i>ifaceHandle</i>	Interface handle.
in	<i>deviceID</i>	Device ID string, can be get from TY_DEVICE_BASE_INFO .
out	<i>deviceHandle</i>	Handle of opened device.

Return values

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

Return values

<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	Invalid interface handle.
<i>TY_STATUS_NULL_POINTER</i>	deviceId or deviceHandle is NULL.
<i>TY_STATUS_INVALID_PARAMETER</i>	Device not found.
<i>TY_STATUS_BUSY</i>	Device has been opened.
<i>TY_STATUS_DEVICE_ERROR</i>	Open device failed.

5.1.5.39 TYOpenDeviceWithIP()

```

TY_CAPI TYOpenDeviceWithIP (
    TY_INTERFACE_HANDLE ifaceHandle,
    const char * IP,
    TY_DEV_HANDLE * deviceHandle )

```

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

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	Invalid interface handle.
<i>TY_STATUS_NULL_POINTER</i>	IP or deviceHandle is NULL.
<i>TY_STATUS_INVALID_PARAMETER</i>	Device not found.
<i>TY_STATUS_BUSY</i>	Device has been opened, may occupied somewhere else.
<i>TY_STATUS_DEVICE_ERROR</i>	Open device failed.

5.1.5.40 TYOpenInterface()

```

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

```

Open specified interface.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_NULL_POINTER</i>	ifaceID or outHandle is NULL.
<i>TY_STATUS_INVALID_INTERFACE</i>	Interface not found.

See also

[TYGetInterfaceList](#)

5.1.5.41 TYRegisterEventCallback()

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

Register device status callback. Register NULL to clean callback.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_BUSY</i>	Device is capturing.

5.1.5.42 TYSendSoftTrigger()

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

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

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_FEATURE</i>	Not support soft trigger.
<i>TY_STATUS_IDLE</i>	Device has not started capture.
<i>TY_STATUS_WRONG_MODE</i>	Not in trigger mode.

5.1.5.43 TYSetBool()

```

TY_CAPI TYSetBool (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    bool value )

```

Set value of bool feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_BOOL.
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.

5.1.5.44 TYSetByteArray()

```

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

```

Write byte array to device.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_BYTEARRAY.
<i>TY_STATUS_NULL_POINTER</i>	pbuffer is NULL.
<i>TY_STATUS_WRONG_SIZE</i>	bufferSize incorrect.
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.

5.1.5.45 TYSetEnum()

```

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

```

Set value of enum feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_ENUM.
<i>TY_STATUS_INVALID_PARAMETER</i>	value is invalid.
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.

5.1.5.46 TYSetFloat()

```

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

```

Set value of float feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_FLOAT.
<i>TY_STATUS_OUT_OF_RANGE</i>	value is out of range.
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.

5.1.5.47 TYSetInt()

```

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

```

Set value of integer feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not <i>TY_FEATURE_INT</i> .
<i>TY_STATUS_OUT_OF_RANGE</i>	value is out of range.
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.

5.1.5.48 TYSetString()

```

TY_CAPI TYSetString (
    TY_DEV_HANDLE hDevice,
    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    const char * buffer )

```

Set value of string feature.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not <i>TY_FEATURE_STRING</i> .
<i>TY_STATUS_NULL_POINTER</i>	buffer is NULL.
<i>TY_STATUS_OUT_OF_RANGE</i>	Input string is too long.
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.

5.1.5.49 TYSetStruct()

```

TY_CAPI TYSetStruct (
    TY_DEV_HANDLE hDevice,

```

```

    TY_COMPONENT_ID componentID,
    TY_FEATURE_ID featureID,
    void * pStruct,
    uint32_t structSize )

```

Set value of struct.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	Invalid component ID.
<i>TY_STATUS_INVALID_FEATURE</i>	Invalid feature ID.
<i>TY_STATUS_NOT_PERMITTED</i>	The feature is not writable.
<i>TY_STATUS_WRONG_TYPE</i>	The feature's type is not TY_FEATURE_STRUCT.
<i>TY_STATUS_NULL_POINTER</i>	pStruct is NULL.
<i>TY_STATUS_WRONG_SIZE</i>	structSize incorrect.
<i>TY_STATUS_BUSY</i>	Device is capturing, the feature is locked.

5.1.5.50 TYStartCapture()

```

TY_CAPI TYStartCapture (
    TY_DEV_HANDLE hDevice )

```

Start capture.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_INVALID_COMPONENT</i>	No components enabled.
<i>TY_STATUS_BUSY</i>	Device has been started.
<i>TY_STATUS_DEVICE_ERROR</i>	Start capture failed.

5.1.5.51 TYStopCapture()

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

Stop capture.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_INVALID_HANDLE</i>	Invalid device handle.
<i>TY_STATUS_IDLE</i>	Device is not capturing.
<i>TY_STATUS_DEVICE_ERROR</i>	Stop capture failed.

5.1.5.52 TYUpdateDeviceList()

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

Update current connected devices.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.
<i>TY_STATUS_INVALID_INTERFACE</i>	Invalid interface handle.

5.1.5.53 TYUpdateInterfaceList()

```
TY_CAPI TYUpdateInterfaceList ( )
```

Update current interfaces. call before TYGetInterfaceList.

Return values

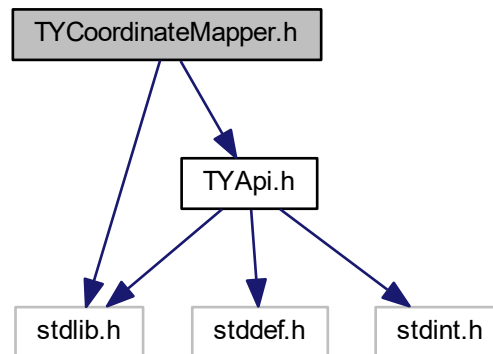
<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NOT_INITED</i>	TYInitLib not called.

5.2 TYCoordinateMapper.h File Reference

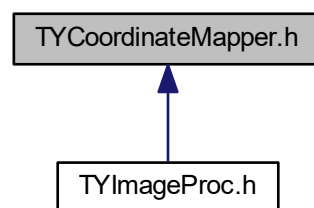
Coordinate Conversion API.

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

Include dependency graph for TYCoordinateMapper.h:



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



Classes

- struct [TY_PIXEL_DESC](#)

Macros

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

Typedefs

- typedef struct [TY_PIXEL_DESC](#) [TY_PIXEL_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)
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)
Map 3D points to pixels on depth image. Reverse operation of TYMapDepthToPoint3d.
- TY_CAPI [TYMapDepthImageToPoint3d](#) (const [TY_CAMERA_CALIB_INFO](#) *src_calib, uint32_t imageW, uint32_t imageH, const uint16_t *depth, [TY_VECT_3F](#) *point3d)
Map depth image to 3D points. 0 depth pixels maps to (NaN, NaN, NaN).
- 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)
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, uint32_t count, [TY_VECT_3F](#) *point3dTo)
Map 3D points to another coordinate.

5.2.1 Detailed Description

Coordinate Conversion API.

Note

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

Copyright

Copyright(C)2016-2018 Percipio All Rights Reserved

5.2.2 Macro Definition Documentation

5.2.2.1 TYMAP_CHECKRET

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

Value:

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

Definition at line 186 of file TYCoordinateMapper.h.

5.2.3 Function Documentation

5.2.3.1 TYInvertExtrinsic()

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

Calculate 4x4 extrinsic matrix's inverse matrix.

Parameters

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

Return values

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

5.2.3.2 TYMapDepthImageToPoint3d()

```
TY_CAPI TYMapDepthImageToPoint3d (
    const TY_CAMERA_CALIB_INFO * src_calib,
    uint32_t imageW,
    uint32_t imageH,
    const uint16_t * depth,
    TY_VECT_3F * point3d )
```

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

Parameters

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

Return values

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

5.2.3.3 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 )
```

Map pixels on depth image to 3D points.

Parameters

in	<i>src_calib</i>	Depth image's calibration data.
in	<i>depthW</i>	Width of depth image.
in	<i>depthH</i>	Height of depth image.
in	<i>depthPixels</i>	Pixels on depth image.
in	<i>count</i>	Number of depth pixels.
out	<i>point3d</i>	Output point3D.

Return values

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

5.2.3.4 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 )
```

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

Parameters

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

Return values

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

5.2.3.5 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 )

```

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

Parameters

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

Return values

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

5.2.3.6 TYMapPoint3dToPoint3d()

```

TY_CAPI TYMapPoint3dToPoint3d (
    const TY_CAMERA_EXTRINSIC * extrinsic,
    const TY_VECT_3F * point3dFrom,
    uint32_t count,
    TY_VECT_3F * point3dTo )

```

Map 3D points to another coordinate.

Parameters

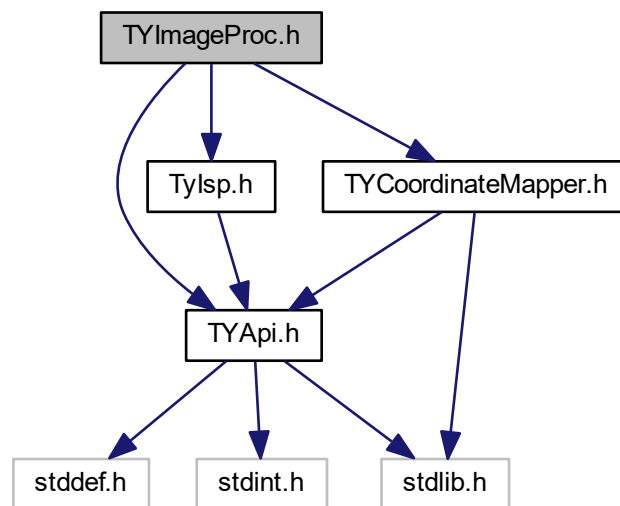
in	<i>extrinsic</i>	Extrinsic matrix.
in	<i>point3dFrom</i>	Source 3D points.
in	<i>count</i>	Number of source 3D points.
out	<i>point3dTo</i>	Target 3D points.

Return values

<code>TY_STATUS_OK</code>	Succeed.
---------------------------	----------

5.3 TYImageProc.h File Reference

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



Classes

- struct [DepthSpeckleFilterParameters](#)
default parameter value definition
- struct [DepthEnhenceParameters](#)
default parameter value definition

Macros

- `#define DepthSpeckleFilterParameters_Initializer {150, 64}`
- `#define DepthEnhenceParameters_Initializer {10, 20, 10, 0.1f}`

Functions

- TY_CAPI [TYUndistortImage](#) (const [TY_CAMERA_CALIB_INFO](#) *srcCalibInfo, const [TY_IMAGE_DATA](#) *srcImage, const [TY_CAMERA_INTRINSIC](#) *cameraNewIntrinsic, [TY_IMAGE_DATA](#) *dstImage)
Do image undistortion, only support TY_PIXEL_FORMAT_MONO, TY_PIXEL_FORMAT_RGB, TY_PIXEL_FORMAT_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.

5.3.1 Detailed Description

Image post-process API

Copyright

Copyright(C)2016-2018 Percipio All Rights Reserved

5.3.2 Function Documentation

5.3.2.1 TYDepthEnhenceFilter()

```
TY_CAPI TYDepthEnhenceFilter (
    const TY\_IMAGE\_DATA * depthImages,
    int imageNum,
    TY\_IMAGE\_DATA * guide,
    TY\_IMAGE\_DATA * output,
    const DepthEnhenceParameters * param )
```

Remove speckles on depth image.

Parameters

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

Return values

<i>TY_STATUS_OK</i>	Succeed.
<i>TY_STATUS_NULL_POINTER</i>	Any depthImage, param, output or output->buffer is NULL.
<i>TY_STATUS_INVALID_PARAMETER</i>	imageNum >= 5 or imageNum <= 0, or any image invalid

Return values

<code>TY_STATUS_OUT_OF_MEMORY</code>	Output image not suitable.
--------------------------------------	----------------------------

5.3.2.2 TYDepthSpeckleFilter()

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

Remove speckles on depth image.

Parameters

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

Return values

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

5.3.2.3 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_FORMAT_BGR.

Parameters

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

Return values

<code>TY_STATUS_OK</code>	Succeed.
---------------------------	----------

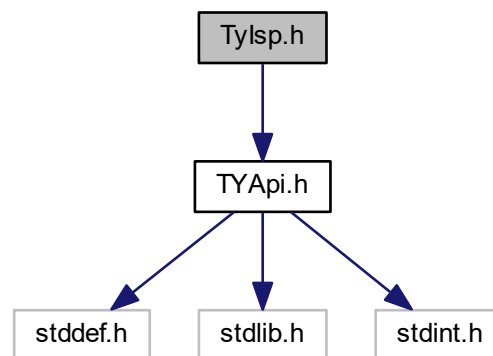
Return values

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

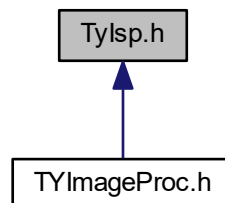
5.4 TyIspp.h File Reference

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

Include dependency graph for TyIspp.h:



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



Classes

- struct [TY_ISP_FEATURE_INFO](#)

Macros

- `#define TYISP_CAPI TY_CAPI`

Typedefs

- `typedef void * TY_ISP_HANDLE`

Enumerations

- enum `TY_ISP_FEATURE_ID` {
`TY_ISP_FEATURE_CAM_MODEL` = 0x000000, `TY_ISP_FEATURE_CAM_DEV_HANDLE` = 0x000001,
`TY_ISP_FEATURE_CAM_DEV_COMPONENT` = 0x000002, `TY_ISP_FEATURE_IMAGE_SIZE` = 0x000100,
`TY_ISP_FEATURE_WHITEBALANCE_GAIN` = 0x000200, `TY_ISP_FEATURE_ENABLE_AUTO_WHITEBALANCE` = 0x000300, `TY_ISP_FEATURE_SHADING` = 0x000400, `TY_ISP_FEATURE_SHADING_CENTER` = 0x000500,
`TY_ISP_FEATURE_BLACK_LEVEL` = 0x000600, `TY_ISP_FEATURE_BLACK_LEVEL_COLUMN` = 0x000610, `TY_ISP_FEATURE_BLACK_LEVEL_GAIN` = 0x000700, `TY_ISP_FEATURE_BLACK_LEVEL_GAIN_COLUMN` = 0x000710,
`TY_ISP_FEATURE_BAYER_PATTERN` = 0x000800, `TY_ISP_FEATURE_DEMOSAIC_METHOD` = 0x000900, `TY_ISP_FEATURE_GAMMA` = 0x000A00, `TY_ISP_FEATURE_DEFECT_PIXEL_LIST` = 0x000B00,
`TY_ISP_FEATURE_CCM` = 0x000C00, `TY_ISP_FEATURE_CCM_ENABLE` = 0x000C10, `TY_ISP_FEATURE_AUTOBRIGHT` = 0x000D00, `TY_ISP_FEATURE_CONTRAST` = 0x000E00,
`TY_ISP_FEATURE_AUTOBRIGHT` = 0x000F00, `TY_ISP_FEATURE_INPUT_RESAMPLE_SCALE` = 0x001000, `TY_ISP_FEATURE_ENABLE_AUTO_EXPOSURE_GAIN` = 0x001100, `TY_ISP_FEATURE_AUTO_EXPOSURE_UPDATE_INTERVAL` = 0x001200,
`TY_ISP_FEATURE_AUTO_GAIN_RANGE` = 0x001300, `TY_ISP_FEATURE_AUTO_EXPOSURE_UPDATE_INTERVAL` = 0x001400, `TY_ISP_FEATURE_DEBUG_LOG` = 0xff000000 }
- enum `TY_ISP_BAYER_PATTERN` {
`TY_ISP_BAYER_GB` = 0, `TY_ISP_BAYER_BG` = 1, `TY_ISP_BAYER_RG` = 2, `TY_ISP_BAYER_GR` = 3,
`TY_ISP_BAYER_AUTO` = 0xff }
- enum `TY_DEMOSAIC_METHOD` { `TY_DEMOSAIC_METHOD_SIMPLE` = 0, `TY_DEMOSAIC_METHOD_BILINEAR` = 1, `TY_DEMOSAIC_METHOD_HQLINEAR` = 2, `TY_DEMOSAIC_METHOD_EDGESENSE` = 3 }

Functions

- `TYISP_CAPI TYISPCreate` (`TY_ISP_HANDLE` *handle)
- `TYISP_CAPI TYISPRelease` (`TY_ISP_HANDLE` *handle)
- `TYISP_CAPI TYISPLoadConfig` (`TY_ISP_HANDLE` handle, const uint8_t *config, uint32_t config_size)
- `TYISP_CAPI TYISPUpdateDevice` (`TY_ISP_HANDLE` handle)
called by main thread to update & control device status for ISP
- `TYISP_CAPI TYISPSetFeature` (`TY_ISP_HANDLE` handle, `TY_ISP_FEATURE_ID` feature_id, const uint8_t *data, int32_t size)
- `TYISP_CAPI TYISPGetFeature` (`TY_ISP_HANDLE` handle, `TY_ISP_FEATURE_ID` feature_id, uint8_t *data_buff, int32_t buff_size)
- `TYISP_CAPI TYISPGetFeatureSize` (`TY_ISP_HANDLE` handle, `TY_ISP_FEATURE_ID` feature_id, int32_t *size)
- `TYISP_CAPI TYISPHasFeature` (`TY_ISP_HANDLE` handle, `TY_ISP_FEATURE_ID` feature_id)
- `TYISP_CAPI TYISPGetFeatureInfoList` (`TY_ISP_HANDLE` handle, `TY_ISP_FEATURE_INFO` *info_buffer, int buffer_size)
- `TYISP_CAPI TYISPGetFeatureInfoListSize` (`TY_ISP_HANDLE` handle, int32_t *buffer_size)
- `TYISP_CAPI TYISPProcessImage` (`TY_ISP_HANDLE` handle, const `TY_IMAGE_DATA` *image_bayer, `TY_IMAGE_DATA` *image_out)
convert bayer raw image to rgb image,output buffer is allocated by invoker

5.4.1 Detailed Description

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

Copyright(C)2016-2019 Percipio All Rights Reserved

5.4.2 Enumeration Type Documentation

5.4.2.1 TY_ISP_FEATURE_ID

enum [TY_ISP_FEATURE_ID](#)

Enumerator

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

Definition at line 17 of file TyIspp.h.

Index

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

TY_CAMERA_CALIB_INFO, [8](#)
 TYApi.h, [29](#)
TY_CAMERA_DISTORTION, [9](#)
TY_CAMERA_EXTRINSIC, [9](#)
 TYApi.h, [29](#)
TY_CAMERA_INTRINSIC, [10](#)
 TYApi.h, [30](#)
TY_CAMERA_STATISTICS, [10](#)
TY_COMPONENT_ID
 TYApi.h, [30](#)
TY_DECLARE_IMAGE_MODE1
 TYApi.h, [29](#)
TY_DEVICE_BASE_INFO, [11](#)
 TYApi.h, [30](#)
TY_DEVICE_COMPONENT_LIST
 TYApi.h, [30](#), [32](#)
TY_DEVICE_NET_INFO, [12](#)
TY_DEVICE_USB_INFO, [12](#)
TY_ENUM_ENTRY, [12](#)
 TYApi.h, [31](#)
TY_EVENT_INFO, [13](#)
TY_FEATURE_ID_LIST
 TYApi.h, [33](#)
TY_FEATURE_INFO, [13](#)
TY_FEATURE_ID
 TYApi.h, [31](#)
TY_FLOAT_RANGE, [14](#)
TY_FRAME_DATA, [14](#)
TY_IMAGE_DATA, [15](#)
TY_INT_RANGE, [16](#)
TY_INTERFACE_INFO, [16](#)
 TYApi.h, [31](#)
TY_ISP_FEATURE_INFO, [17](#)
TY_ISP_FEATURE_ID
 Tylsp.h, [75](#)
TY_PIXEL_DESC, [17](#)
TY_PIXEL_FORMAT_LIST
 TYApi.h, [34](#)
TY_RESOLUTION_MODE_LIST
 TYApi.h, [34](#)
TY_TRIGGER_ACTIVATION_LIST
 TYApi.h, [31](#), [35](#)
TY_TRIGGER_MODE_LIST
 TYApi.h, [32](#), [35](#)
TY_TRIGGER_PARAM_EX, [18](#)
TY_TRIGGER_PARAM, [18](#)
TY_VECT_3F, [19](#)

TY_VERSION_INFO, [19](#)
TYApi.h, [21](#)
 TY_CAMERA_CALIB_INFO, [29](#)
 TY_CAMERA_EXTRINSIC, [29](#)
 TY_CAMERA_INTRINSIC, [30](#)
 TY_COMPONENT_ID, [30](#)
 TY_DECLARE_IMAGE_MODE1, [29](#)
 TY_DEVICE_BASE_INFO, [30](#)
 TY_DEVICE_COMPONENT_LIST, [30](#), [32](#)
 TY_ENUM_ENTRY, [31](#)
 TY_FEATURE_ID_LIST, [33](#)
 TY_FEATURE_ID, [31](#)
 TY_INTERFACE_INFO, [31](#)
 TY_PIXEL_FORMAT_LIST, [34](#)
 TY_RESOLUTION_MODE_LIST, [34](#)
 TY_TRIGGER_ACTIVATION_LIST, [31](#), [35](#)
 TY_TRIGGER_MODE_LIST, [32](#), [35](#)
TYClearBufferQueue, [35](#)
TYCloseDevice, [36](#)
TYCloseInterface, [36](#)
TYDeinitLib, [37](#)
TYDisableComponents, [37](#)
TYEnableComponents, [37](#)
TYEnqueueBuffer, [38](#)
TYErrorString, [38](#)
TYFetchFrame, [39](#)
TYForceDeviceIP, [39](#)
TYGetBool, [40](#)
TYGetByteArray, [40](#)
TYGetByteArraySize, [42](#)
TYGetComponentIDs, [42](#)
TYGetDeviceInfo, [43](#)
TYGetDeviceInterface, [43](#)
TYGetDeviceList, [44](#)
TYGetDeviceNumber, [44](#)
TYGetEnabledComponents, [46](#)
TYGetEnum, [46](#)
TYGetEnumEntryCount, [47](#)
TYGetEnumEntryInfo, [47](#)
TYGetFeatureInfo, [48](#)
TYGetFloat, [49](#)
TYGetFloatRange, [49](#)
TYGetFrameBufferSize, [50](#)
TYGetInt, [50](#)
TYGetIntRange, [52](#)
TYGetInterfaceList, [51](#)
TYGetInterfaceNumber, [51](#)
TYGetString, [52](#)
TYGetStringLength, [53](#)

- TYGetStruct, [54](#)
- TYHasDevice, [54](#)
- TYHasFeature, [55](#)
- TYHasInterface, [55](#)
- TYLibVersion, [56](#)
- TYOpenDevice, [56](#)
- TYOpenDeviceWithIP, [57](#)
- TYOpenInterface, [57](#)
- TYRegisterEventCallback, [58](#)
- TYSendSoftTrigger, [58](#)
- TYSetBool, [59](#)
- TYSetByteArray, [59](#)
- TYSetEnum, [60](#)
- TYSetFloat, [61](#)
- TYSetInt, [61](#)
- TYSetString, [62](#)
- TYSetStruct, [62](#)
- TYStartCapture, [63](#)
- TYStopCapture, [63](#)
- TYUpdateDeviceList, [64](#)
- TYUpdateInterfaceList, [64](#)
- TYClearBufferQueue
 - TYApi.h, [35](#)
- TYCloseDevice
 - TYApi.h, [36](#)
- TYCloseInterface
 - TYApi.h, [36](#)
- TYCoordinateMapper.h, [65](#)
 - TYInvertExtrinsic, [67](#)
 - TYMAP_CHECKRET, [66](#)
 - TYMapDepthImageToPoint3d, [67](#)
 - TYMapDepthToPoint3d, [67](#)
 - TYMapPoint3dToDepth, [68](#)
 - TYMapPoint3dToDepthImage, [69](#)
 - TYMapPoint3dToPoint3d, [69](#)
- TYDeinitLib
 - TYApi.h, [37](#)
- TYDepthEnhanceFilter
 - TYImageProc.h, [71](#)
- TYDepthSpeckleFilter
 - TYImageProc.h, [72](#)
- TYDisableComponents
 - TYApi.h, [37](#)
- TYEnableComponents
 - TYApi.h, [37](#)
- TYEnqueueBuffer
 - TYApi.h, [38](#)
- TYErrorString
 - TYApi.h, [38](#)
- TYFetchFrame
 - TYApi.h, [39](#)
- TYForceDeviceIP
 - TYApi.h, [39](#)
- TYGetBool
 - TYApi.h, [40](#)
- TYGetByteArray
 - TYApi.h, [40](#)
- TYGetByteArraySize
 - TYApi.h, [42](#)
- TYGetComponentIDs
 - TYApi.h, [42](#)
- TYGetDeviceInfo
 - TYApi.h, [43](#)
- TYGetDeviceInterface
 - TYApi.h, [43](#)
- TYGetDeviceList
 - TYApi.h, [44](#)
- TYGetDeviceNumber
 - TYApi.h, [44](#)
- TYGetEnabledComponents
 - TYApi.h, [46](#)
- TYGetEnum
 - TYApi.h, [46](#)
- TYGetEnumEntryCount
 - TYApi.h, [47](#)
- TYGetEnumEntryInfo
 - TYApi.h, [47](#)
- TYGetFeatureInfo
 - TYApi.h, [48](#)
- TYGetFloat
 - TYApi.h, [49](#)
- TYGetFloatRange
 - TYApi.h, [49](#)
- TYGetFrameBufferSize
 - TYApi.h, [50](#)
- TYGetInt
 - TYApi.h, [50](#)
- TYGetIntRange
 - TYApi.h, [52](#)
- TYGetInterfaceList
 - TYApi.h, [51](#)
- TYGetInterfaceNumber
 - TYApi.h, [51](#)
- TYGetString
 - TYApi.h, [52](#)
- TYGetStringLength
 - TYApi.h, [53](#)
- TYGetStruct
 - TYApi.h, [54](#)
- TYHasDevice
 - TYApi.h, [54](#)
- TYHasFeature
 - TYApi.h, [55](#)
- TYHasInterface
 - TYApi.h, [55](#)
- TYImageProc.h, [70](#)
 - TYDepthEnhanceFilter, [71](#)
 - TYDepthSpeckleFilter, [72](#)
 - TYUndistortImage, [72](#)
- TYInvertExtrinsic
 - TYCoordinateMapper.h, [67](#)
- TYLibVersion
 - TYApi.h, [56](#)
- TYMAP_CHECKRET
 - TYCoordinateMapper.h, [66](#)
- TYMapDepthImageToPoint3d

- TYCoordinateMapper.h, [67](#)
- TYMapDepthToPoint3d
 - TYCoordinateMapper.h, [67](#)
- TYMapPoint3dToDepth
 - TYCoordinateMapper.h, [68](#)
- TYMapPoint3dToDepthImage
 - TYCoordinateMapper.h, [69](#)
- TYMapPoint3dToPoint3d
 - TYCoordinateMapper.h, [69](#)
- TYOpenDevice
 - TYApi.h, [56](#)
- TYOpenDeviceWithIP
 - TYApi.h, [57](#)
- TYOpenInterface
 - TYApi.h, [57](#)
- TYRegisterEventCallback
 - TYApi.h, [58](#)
- TYSendSoftTrigger
 - TYApi.h, [58](#)
- TYSetBool
 - TYApi.h, [59](#)
- TYSetByteArray
 - TYApi.h, [59](#)
- TYSetEnum
 - TYApi.h, [60](#)
- TYSetFloat
 - TYApi.h, [61](#)
- TYSetInt
 - TYApi.h, [61](#)
- TYSetString
 - TYApi.h, [62](#)
- TYSetStruct
 - TYApi.h, [62](#)
- TYStartCapture
 - TYApi.h, [63](#)
- TYStopCapture
 - TYApi.h, [63](#)
- TYUndistortImage
 - TYImageProc.h, [72](#)
- TYUpdateDeviceList
 - TYApi.h, [64](#)
- TYUpdateInterfaceList
 - TYApi.h, [64](#)
- TyIsp.h, [73](#)
 - TY_ISP_FEATURE_ID, [75](#)