Percipio Camport SDK 2.6.3

Generated by Doxygen 1.8.11

Contents

1	Clas	s Index	1
	1.1	Class List	1
2	File	Index	3
	2.1	File List	3
3	Clas	es Documentation	5
	3.1	TY_CAMERA_DISTORTION Struct Reference	5
		3.1.1 Detailed Description	5
	3.2	TY_CAMERA_EXTRINSIC Struct Reference	5
		3.2.1 Detailed Description	6
	3.3	TY_CAMERA_INTRINSIC Struct Reference	6
		3.3.1 Detailed Description	6
	3.4	TY_DEVICE_BASE_INFO Struct Reference	6
		3.4.1 Detailed Description	7
	3.5	TY_DEVICE_NET_INFO Struct Reference	7
		3.5.1 Detailed Description	7
	3.6	TY_ENUM_ENTRY Struct Reference	7
		3.6.1 Detailed Description	7
	3.7	TY_EVENT_INFO Struct Reference	8
		3.7.1 Detailed Description	8
	3.8	TY_FEATURE_INFO Struct Reference	8
		3.8.1 Detailed Description	8
	3 9	TV_FLOAT_BANGE Struct Reference	q

iv CONTENTS

		3.9.1	Detailed	Description	9
	3.10	TY_FF	RAME_DA	TA Struct Reference	9
		3.10.1	Detailed	Description	9
	3.11	TY_IM	AGE_DAT	A Struct Reference	10
		3.11.1	Detailed	Description	10
	3.12	TY_IN	T_RANGE	Struct Reference	10
		3.12.1	Detailed	Description	10
	3.13	TY_TF	RIGGER_M	MODE Struct Reference	11
		3.13.1	Detailed	Description	11
	3.14	TY_VE	CT_3F St	ruct Reference	11
		3.14.1	Detailed	Description	11
	3.15	TY_VE	RSION_II	NFO Struct Reference	11
		3.15.1	Detailed	Description	11
4	Eile I	Dooum	entation		13
4				eference	
	4.1	_			13
		4.1.1		Description	18
		4.1.2		ation Type Documentation	19
			4.1.2.1	TY_DEVICE_COMPONENT_LIST	19
			4.1.2.2	TY_FEATURE_ID_LIST	
			4.1.2.3	TY_IMAGE_MODE_LIST	20
		4.1.3		Documentation	20
			4.1.3.1	TYClearBufferQueue(TY_DEV_HANDLE hDevice)	20
			4.1.3.2	TYCloseDevice(TY_DEV_HANDLE hDevice)	20
			4.1.3.3	TYDeinitLib(void)	20
			4.1.3.4	$\label{thm:const} \begin{split} & TYDepthToWorld(TY_DEV_HANDLE\ hDevice,\ const\ TY_VECT_3F\ *depth,\ T \leftrightarrow Y_VECT_3F\ *world,\ int32_t\ worldPaddingBytes,\ int32_t\ pointCount) \\ & \dots \dots \\ \end{split}$	21
			4.1.3.5	TYDisableComponents(TY_DEV_HANDLE hDevice, int32_t componentIDs)	21
			4.1.3.6	TYEnableComponents(TY_DEV_HANDLE hDevice, int32_t componentIDs)	22
			4.1.3.7	TYEnqueueBuffer(TY_DEV_HANDLE hDevice, void *buffer, int32_t bufferSize) .	22
			4.1.3.8	TYEnterDeveloperMode(TY_DEV_HANDLE hDevice)	22

CONTENTS

4.1.3.9	TYErrorString(TY_STATUS errorID)	23
4.1.3.10	TYFetchFrame(TY_DEV_HANDLE hDevice, TY_FRAME_DATA *frame, int32← _t timeout)	23
4.1.3.11	TYGetBool(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, bool *value)	23
4.1.3.12	TYGetComponentIDs(TY_DEV_HANDLE hDevice, int32_t *componentIDs)	24
4.1.3.13	TYGetDeviceInfo(TY_DEV_HANDLE hDevice, TY_DEVICE_BASE_INFO *info)	24
4.1.3.14	TYGetDeviceList(TY_DEVICE_BASE_INFO *deviceInfos, int32_t bufferCount, int32_t *filledDeviceCount)	24
4.1.3.15	TYGetDeviceNumber(int32_t *deviceNumber)	25
4.1.3.16	$\label{thm:component} \begin{split} & TYGetEnabledComponentIDs(TY_DEV_HANDLE\ hDevice,\ int 32_t\ *component \mapsto IDs)\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\$	25
4.1.3.17	TYGetEnum(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentI ← D, TY_FEATURE_ID featureID, int32_t *value)	25
4.1.3.18	$\label{thm:cont} \begin{split} & TYGetEnumEntryCount(TY_DEV_HANDLE hDevice, TY_COMPONENT_I \leftrightarrow \\ & D \ componentID, \ TY_FEATURE_ID \ featureID, \ int32_t \ *entryCount) \\ & \ldots \ldots \\ \end{split}$	26
4.1.3.19	TYGetEnumEntryInfo(TY_DEV_HANDLE hDevice, TY_COMPONENT_I↔ D componentID, TY_FEATURE_ID featureID, TY_ENUM_ENTRY *entries, int32_t entryCount, int32_t *filledEntryCount)	26
4.1.3.20	TYGetFeatureInfo(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID component← ID, TY_FEATURE_ID featureID, TY_FEATURE_INFO *featureInfo)	
4.1.3.21	TYGetFloat(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentI ← D, TY_FEATURE_ID featureID, float *value)	27
4.1.3.22	TYGetFloatRange(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID component LD, TY_FEATURE_ID featureID, TY_FLOAT_RANGE *floatRange)	_ 28
4.1.3.23	TYGetFrameBufferSize(TY_DEV_HANDLE hDevice, int32_t *bufferSize)	28
4.1.3.24	TYGetInt(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, T← Y_FEATURE_ID featureID, int32_t ∗value)	28
4.1.3.25	TYGetIntRange(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID component ← ID, TY_FEATURE_ID featureID, TY_INT_RANGE ∗intRange)	29
4.1.3.26	TYGetString(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentI ← D, TY_FEATURE_ID featureID, char *buffer, int32_t bufferSize)	29
4.1.3.27	$\label{thm:component_loss} \begin{split} & TYGetStringBufferSize(TY_DEV_HANDLE hDevice, TY_COMPONENT_I & \mapsto \\ & D \ componentID, \ TY_FEATURE_ID \ featureID, \ int32_t \ *size) \\ & \cdot \cdot$	30
4.1.3.28	TYGetStruct(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentI ← D, TY_FEATURE_ID featureID, void *pStruct, int32_t structSize)	30
4.1.3.29	TYIsCapturing(TY_DEV_HANDLE hDevice, bool *isCapturing)	31
4.1.3.30	TYLibVersion(TY VERSION INFO *version)	31

vi

4.1.3.31	TYOpenDevice(const char *deviceID, TY_DEV_HANDLE *deviceHandle)	31
4.1.3.32	TYOpenDeviceWithIP(const char *IP, TY_DEV_HANDLE *deviceHandle)	32
4.1.3.33	TYRegisterCallback(TY_DEV_HANDLE hDevice, TY_FRAME_CALLBACK callback, void *userdata)	32
4.1.3.34	TYRegisterEventCallback(TY_DEV_HANDLE hDevice, TY_EVENT_CALLBA⇔ CK callback, void *userdata)	33
4.1.3.35	TYSendSoftTrigger(TY_DEV_HANDLE hDevice)	33
4.1.3.36	TYSetBool(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, bool value)	33
4.1.3.37	TYSetEnum(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentI ← D, TY_FEATURE_ID featureID, int32_t value)	34
4.1.3.38	TYSetFloat(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, float value)	34
4.1.3.39	TYSetInt(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, T↔ Y_FEATURE_ID featureID, int32_t value)	35
4.1.3.40	TYSetString(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentI ← D, TY_FEATURE_ID featureID, const char *buffer)	35
4.1.3.41	TYSetStruct(TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentI ← D, TY_FEATURE_ID featureID, void *pStruct, int32_t structSize)	36
4.1.3.42	TYStartCapture(TY_DEV_HANDLE hDevice)	36
4.1.3.43	TYStopCapture(TY_DEV_HANDLE hDevice)	37
4.1.3.44	TYUndistortImage(const TY_CAMERA_INTRINSIC *cameraIntrinsic, const T← Y_CAMERA_DISTORTION *cameraDistortion, const TY_CAMERA_INTRINSI← C *cameraNewIntrinsic, const TY_IMAGE_DATA *srcImage, TY_IMAGE_DATA *dstImage)	37
4.1.3.45	TYWorldToDepth(TY_DEV_HANDLE hDevice, const TY_VECT_3F *world, TY ← _ VECT_3F *depth, int32_t worldPaddingBytes, int32_t pointCount)	38
Index		39

Chapter 1

Class Index

1.1 Class List

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

TY_CAMERA_DISTORTION
Camera distortion parameters
TY_CAMERA_EXTRINSIC
TY_CAMERA_INTRINSIC
TY_DEVICE_BASE_INFO 6
TY_DEVICE_NET_INFO
TY_ENUM_ENTRY
TY_EVENT_INFO 8
TY_FEATURE_INFO 8
TY_FLOAT_RANGE 9
TY_FRAME_DATA 9
TY_IMAGE_DATA 10
TY_INT_RANGE 10
TY_TRIGGER_MODE
TY_VECT_3F
TY_VERSION_INFO

2 Class Index

Chapter 2

File Index

A 4			
ר כי	Ηı	1 1	ct
Z . I		_	Э1

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

TY_API.h											
API for Percipio depth cameras	 		 	 		 	 				13

File Index

Chapter 3

Class Documentation

3.1 TY_CAMERA_DISTORTION Struct Reference

camera distortion parameters

```
#include <TY_API.h>
```

Public Attributes

• float data [12] k1,k2,p1,p2,k3,k4,k5,k6,s1,s2,s3,s4

3.1.1 Detailed Description

camera distortion parameters

Definition at line 404 of file TY_API.h.

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

• TY_API.h

3.2 TY_CAMERA_EXTRINSIC Struct Reference

```
#include <TY_API.h>
```

Public Attributes

• float data [4 *4]

6 Class Documentation

3.2.1 Detailed Description

```
[r11, r12, r13, t1, r21, r22, r23, t2, r31, r32, r33, t3, 0, 0, 0, 1]
```

Definition at line 398 of file TY_API.h.

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

• TY_API.h

3.3 TY_CAMERA_INTRINSIC Struct Reference

```
#include <TY_API.h>
```

Public Attributes

• float data [3 *3]

3.3.1 Detailed Description

```
[fx, 0, cx, 0, fy, cy, 0, 0, 1]
```

Definition at line 389 of file TY_API.h.

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

• TY_API.h

3.4 TY_DEVICE_BASE_INFO Struct Reference

Public Attributes

- TY_INTERFACE devInterface interface, see TY_INTERFACE_LIST
- char id [32]
- char vendorName [32]
- char modelName [32]
- TY_VERSION_INFO hardwareVersion
- TY_VERSION_INFO firmwareVersion
- TY_DEVICE_NET_INFO netInfo
- TY_STATUS status
- char reserved [248]

3.4.1 Detailed Description

Definition at line 329 of file TY_API.h.

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

• TY_API.h

3.5 TY_DEVICE_NET_INFO Struct Reference

Public Attributes

- char mac [32]
- char ip [32]
- · char netmask [32]
- · char gateway [32]
- char reserved [256]

3.5.1 Detailed Description

Definition at line 320 of file TY_API.h.

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

• TY_API.h

3.6 TY_ENUM_ENTRY Struct Reference

Public Attributes

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

3.6.1 Detailed Description

Definition at line 372 of file TY_API.h.

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

8 Class Documentation

3.7 TY_EVENT_INFO Struct Reference

Public Attributes

TY_EVENT eventId

3.7.1 Detailed Description

Definition at line 446 of file TY_API.h.

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

• TY_API.h

3.8 TY_FEATURE_INFO Struct Reference

Public Attributes

bool isValid

true if feature exists, false otherwise

• int8_t accessMode

feature access mode, see TY_ACCESS_MODE_LIST

· bool writableAtRun

feature can be written while capturing

- char reserved0 [1]
- TY_COMPONENT_ID componentID
- TY_FEATURE_ID featureID
- char **name** [32]
- int32_t bindComponentID

component ID current feature bind to

• int32_t bindFeatureID

feature ID current feature bind to

· char reserved [252]

3.8.1 Detailed Description

Definition at line 342 of file TY_API.h.

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

3.9 TY_FLOAT_RANGE Struct Reference

Public Attributes

- · float min
- float max
- float inc
- float reserved [1]

3.9.1 Detailed Description

Definition at line 364 of file TY_API.h.

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

• TY_API.h

3.10 TY_FRAME_DATA Struct Reference

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.

3.10.1 Detailed Description

Definition at line 434 of file TY_API.h.

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

10 Class Documentation

3.11 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 [8]

Reserved.

3.11.1 Detailed Description

Definition at line 419 of file TY_API.h.

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

• TY_API.h

3.12 TY_INT_RANGE Struct Reference

Public Attributes

- int32 t min
- int32_t max
- int32_t inc
- int32_t reserved [1]

3.12.1 Detailed Description

Definition at line 356 of file TY API.h.

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

3.13 TY_TRIGGER_MODE Struct Reference

Public Attributes

- int16_t mode
- int8_t fps

3.13.1 Detailed Description

Definition at line 409 of file TY_API.h.

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

• TY_API.h

3.14 TY_VECT_3F Struct Reference

Public Attributes

- float x
- · float y
- float z

3.14.1 Detailed Description

Definition at line 379 of file TY_API.h.

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

• TY_API.h

3.15 TY_VERSION_INFO Struct Reference

Public Attributes

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

3.15.1 Detailed Description

Definition at line 312 of file TY_API.h.

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

• TY API.h

12 Class Documentation

Chapter 4

File Documentation

4.1 TY_API.h File Reference

API for Percipio depth cameras.

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

Classes

- struct TY_VERSION_INFO
- struct TY_DEVICE_NET_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_TRIGGER_MODE
- 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 2
- #define TY_LIB_VERSION_MINOR 6
- #define TY_LIB_VERSION_PATCH 3
- #define TY_CAPI TY EXTC TY EXPORT TY STATUS TY STDC

Typedefs

- typedef enum TY STATUS LIST TY STATUS LIST
- typedef int32 t TY STATUS
- typedef enum TY_EVENT_LIST TY_ENENT_LIST
- typedef int32 t TY_EVENT
- typedef void * TY DEV HANDLE
- typedef enum TY_DEVICE_COMPONENT_LIST TY_DEVICE_COMPONENT_LIST
- typedef int32 t TY COMPONENT ID
- typedef enum TY_FEATURE_TYPE_LIST TY_FEATURE_TYPE_LIST
- typedef int32_t TY_FEATURE_TYPE
- typedef enum TY FEATURE ID LIST TY FEATURE ID LIST
- typedef int32 t TY_FEATURE_ID
- typedef enum TY_IMAGE_MODE_LIST TY_IMAGE_MODE_LIST
- typedef int32_t TY_IMAGE_MODE
- typedef enum TY TRIGGER ACTIVATION LIST TY TRIGGER ACTIVATION LIST
- typedef int32_t TY_TRIGGER_ACTIVATION
- typedef enum TY_INTERFACE_LIST TY_INTERFACE_LIST
- typedef int32 t TY INTERFACE
- typedef enum TY ACCESS MODE LIST TY ACCESS MODE LIST
- typedef enum TY_PIXEL_TYPE_LIST TY_PIXEL_TYPE_LIST
- typedef enum TY PIXEL BITS LIST TY PIXEL BITS LIST
- typedef enum TY_PIXEL_FORMAT_LIST TY_PIXEL_FORMAT_LIST
- typedef int32_t TY_PIXEL_FORMAT
- typedef enum TY_TRIGGER_MODE_LIST TY_TRIGGER_MODE_LIST
- typedef struct TY_VERSION_INFO TY_VERSION_INFO
- typedef struct TY DEVICE NET INFO TY DEVICE NET 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_TRIGGER_MODE TY_TRIGGER_MODE
- typedef struct TY_IMAGE_DATA TY_IMAGE_DATA
- typedef struct TY_FRAME_DATA TY_FRAME_DATA
- typedef void(* TY_FRAME_CALLBACK) (TY_FRAME_DATA *, void *userdata)
- 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 } • enum TY_EVENT_LIST { TY_EVENT_DEVICE_OFFLINE = -2001 } enum TY DEVICE COMPONENT_LIST { TY COMPONENT DEVICE = 0x80000000, TY COMPONENT DEPTH CAM = 0x00010000, TY COMP↔ ONENT POINT3D CAM = 0x00020000, TY COMPONENT IR CAM LEFT = 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 COMPO NENT_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 F↔ **EATURE BOOL** = 0x4000, TY FEATURE STRING = 0x5000, TY FEATURE BYTEARRAY = 0x6000, TY FEATURE STRUCT = 0x7000 } enum TY FEATURE ID LIST { TY STRUCT CAM INTRINSIC = 0x000 | TY FEATURE STRUCT, TY STRUCT EXTRINSIC TO LEF ← T IR = 0x001 | TY FEATURE STRUCT, TY STRUCT EXTRINSIC TO LEFT RGB = 0x002 | TY FEA↔ TURE STRUCT, TY STRUCT NET INFO = 0x005 | TY FEATURE STRUCT, TY_STRUCT_CAM_DISTORTION = 0x006 | TY_FEATURE_STRUCT, TY_INT_WIDTH_MAX = 0x100 | T Y_FEATURE_INT, TY_INT_HEIGHT_MAX = 0x101 | TY_FEATURE_INT, TY_INT_OFFSET_X = 0x102 | TY FEATURE INT. TY INT OFFSET Y = 0x103 | TY FEATURE INT, TY INT WIDTH = 0x104 | TY FEATURE INT, TY IN ↔ T_HEIGHT = 0x105 | TY_FEATURE_INT, TY_INT_IMAGE_SIZE = 0x106 | TY_FEATURE_INT, TY ENUM PIXEL FORMAT = 0x107 | TY FEATURE ENUM, TY ENUM IMAGE MODE = 0x108 | TY↔ FEATURE ENUM, TY BOOL TRIGGER MODE = 0x200 | TY FEATURE BOOL, TY ENUM TRIGGE ← R_ACTIVATION = 0x201 | TY_FEATURE_ENUM, TY_INT_FRAME_PER_TRIGGER = 0x202 | TY_FEATURE_INT, TY_BOOL_AUTO_EXPOSURE = 0x300 | TY_FEATURE_BOOL, TY_INT_EXPOSURE_TIME = 0x301 | TY_FEATURE_INT, TY_BOOL_AUTO_GAIN = 0x302 | TY FEATURE BOOL, TY_INT_GAIN = 0x303 | TY_FEATURE_INT, TY_BOOL_AUTO_AWB = 0x304 | TY_FEATURE_BOOL, T↔ Y_INT_LASER_POWER = 0x500 | TY_FEATURE_INT, TY_BOOL_LASER_AUTO_CTRL = 0x501 | TY_F↔ EATURE BOOL, TY BOOL UNDISTORTION = 0x510 | TY FEATURE BOOL TY BOOL BRIGHTNESS HISTOGRAM = 0x511 | TY FEATURE BOOL, TY INT R GAIN = 0x520 | TY FEATURE INT, TY INT G GAIN = 0x521 | TY FEATURE INT, TY INT B GAIN = 0x522 | TY FEATURE INT, TY STRUCT WORK MODE = 0x523 | TY FEATURE ← STRUCT } enum TY_IMAGE_MODE_LIST { TY_IMAGE_MODE_320x240 = (320<<12)+240, TY_IMAGE_MODE ← $_{640 \times 480} = (640 < <12) + 480$, TY_IMAGE_MODE_ $_{1280 \times 960} = (1280 < <12) + 960$, TY_IMAGE_MODE_ $_{\hookleftarrow}$ $2592 \times 1944 = (2592 < < 12) + 1944$ • enum TY TRIGGER ACTIVATION LIST { TY TRIGGER ACTIVATION FALLINGEDGE = 0, TY TRIG← **GER ACTIVATION RISINGEDGE** = 1 }

enum TY INTERFACE LIST { TY_INTERFACE UNKNOWN = 0, TY_INTERFACE ETHERNET = 1, TY ←

enum TY ACCESS MODE LIST { TY ACCESS READABLE = 0x1, TY ACCESS WRITABLE = 0x2 }

Generated by Doxygen

INTERFACE USB = 2 }

enum TY_PIXEL_BITS_LIST {

TY_PIXEL_8BIT = 0x00080000, **TY_PIXEL_16BIT** = 0x00100000, **TY_PIXEL_24BIT** = 0x00180000, **TY_\leftrightarrow PIXEL_32BIT** = 0x00200000,

TY PIXEL 96BIT = 0x00600000 }

enum TY PIXEL FORMAT LIST {

 $\begin{array}{l} \textbf{TY_PIXEL_FORMAT_UNDEFINED} = 0, \ \textbf{TY_PIXEL_FORMAT_MONO} = (TY_PIXEL_MONO \mid TY_PIXEL\longleftrightarrow \\ _8BIT \mid 0x00000), \ \textbf{TY_PIXEL_FORMAT_RGB} = (TY_PIXEL_COLOR \mid TY_PIXEL_24BIT \mid 0x0010), \ \textbf{TY_PIXEL_FORMAT_YUV422} = (TY_PIXEL_COLOR \mid TY_PIXEL_16BIT \mid 0x0011), \end{array}$

TY_PIXEL_FORMAT_YVYU = TY_PIXEL_FORMAT_YUV422, TY_PIXEL_FORMAT_YUVV = (TY_PIXE \leftarrow L_COLOR | TY_PIXEL_16BIT | 0x0012), TY_PIXEL_FORMAT_JPEG = (TY_PIXEL_COLOR | TY_PIXE \leftarrow L_24BIT | 0x0013), TY_PIXEL_FORMAT_DEPTH16 = (TY_PIXEL_DEPTH | TY_PIXEL_16BIT | 0x0020), TY_PIXEL_FORMAT_FPOINT3D = (TY_PIXEL_POINT3D | TY_PIXEL_96BIT | 0x0030) }

• enum TY_TRIGGER_MODE_LIST { TY_TRIGGER_MODE_CONTINUES = 0, TY_TRIGGER_MODE_TR ← IG_SLAVE = 1, TY_TRIGGER_MODE_M_SIG = 2, TY_TRIGGER_MODE_M_PER = 3 }

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 TYGetDeviceNumber (int32 t *deviceNumber)

Get number of current connected devices.

TY_CAPI TYGetDeviceList (TY_DEVICE_BASE_INFO *deviceInfos, int32_t bufferCount, int32_t *filled←
 DeviceCount)

Get device info list.

TY_CAPI TYOpenDevice (const char *deviceID, TY_DEV_HANDLE *deviceHandle)
 Open device by device ID.

• TY CAPI TYOpenDeviceWithIP (const char *IP, TY DEV HANDLE *deviceHandle)

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

TY_CAPI TYCloseDevice (TY_DEV_HANDLE hDevice)

Close device by device handle.

• TY_CAPI TYEnterDeveloperMode (TY_DEV_HANDLE hDevice)

Enable developer mode 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 TYGetEnabledComponentIDs (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, int32 t *bufferSize)

Get total buffer size of one frame in current configuration.

• TY CAPI TYEnqueueBuffer (TY DEV HANDLE hDevice, void *buffer, int32 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 TYIsCapturing (TY DEV HANDLE hDevice, bool *isCapturing)

Get if the device is capturing.

TY CAPI TYSendSoftTrigger (TY DEV HANDLE hDevice)

Send a software trigger when device works in trigger mode.

 TY_CAPI TYRegisterCallback (TY_DEV_HANDLE hDevice, TY_FRAME_CALLBACK callback, void *userdata)

Register callback of frame. Register NULL to clean callback.

 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 TYGetFeatureInfo (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEAT
URE ID featureID, TY_FEATURE_INFO *featureInfo)

Get feature info.

TY_CAPI TYGetIntRange (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATU

RE 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_FEA← TURE 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.

Get number of enum entries.

• TY_CAPI TYGetEnumEntryInfo (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_F ← EATURE_ID featureID, TY_ENUM_ENTRY *entries, int32_t entryCount, int32_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 TYGetStringBufferSize (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_←
FEATURE ID featureID, int32 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, int32 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, int32 t structSize)

Get value of struct

• TY_CAPI TYSetStruct (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, void *pStruct, int32 t structSize)

Set value of struct.

TY_CAPI TYDepthToWorld (TY_DEV_HANDLE hDevice, const TY_VECT_3F *depth, TY_VECT_3F *world, int32 t worldPaddingBytes, int32 t pointCount)

TY_CAPI TYWorldToDepth (TY_DEV_HANDLE hDevice, const TY_VECT_3F *world, TY_VECT_3F *depth, int32_t worldPaddingBytes, int32_t pointCount)

TY_CAPI TYUndistortImage (const TY_CAMERA_INTRINSIC *cameraIntrinsic, const TY_CAMERA_DIS
 — TORTION *cameraDistortion, const TY_CAMERA_INTRINSIC *cameraNewIntrinsic, const TY_IMAGE_D
 — ATA *srcImage, TY_IMAGE_DATA *dstImage)

Correct image for lens distortion Format of source image data should be TY_PIXEL_FORMAT_MONO or TY_PIXE ← L_FORMAT_RGB Output buffer is allocated by caller. For IR image undistortion, enable TY_BOOL_UNDISTORTION to get better performance.

- TY_CAPI _TYInitLib (void)
- TY_EXTC TY_EXPORT const char * TYGetFirmwareVer (const char *deviceID)
- TY_CAPI TYGetEnabledComponents (TY_DEV_HANDLE hDevice, int32_t *componentlDs)
- TY_CAPI **TYGetStringLength** (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FE ↔ ATURE_ID featureID, int32_t *length)
- TY_CAPI **TYRegisterWorldToColor** (TY_DEV_HANDLE hDevice, const **TY_VECT_3F** *world, int32_ t worldPaddingBytes, int32 t pointCount, uint16 t *outDepthBuffer, int32 t bufferSize)

4.1.1 Detailed Description

API for Percipio depth cameras.

Copyright(C)2016 Percipio All Rights Reserved

4.1.2 Enumeration Type Documentation

4.1.2.1 enum TY DEVICE COMPONENT LIST

Enumerator

TY_COMPONENT_DEVICE Abstract component stands for whole device, always enabled.

TY_COMPONENT_DEPTH_CAM Depth camera.

TY_COMPONENT_POINT3D_CAM Point3D 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

Definition at line 154 of file TY_API.h.

4.1.2.2 enum TY_FEATURE_ID_LIST

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_EXTRINSIC_TO_LEFT_RGB extrinsic from current component to left RGB, see TY_CAME↔
RA EXTRINSIC

TY_STRUCT_NET_INFO see TY DEVICE NET INFO

TY_STRUCT_CAM_DISTORTION see TY CAMERA DISTORTION

TY_ENUM_PIXEL_FORMAT Pixel format, see TY_PIXEL_FORMAT_LIST.

TY_ENUM_IMAGE_MODE Pixel format, see TY_IMAGE_MODE_LIST.

TY_BOOL_TRIGGER_MODE Trigger mode switch.

TY_ENUM_TRIGGER_ACTIVATION Trigger activation, see TY_TRIGGER_ACTIVATION_LIST.

TY_INT_FRAME_PER_TRIGGER Number of frames captured per trigger.

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 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_INT_R_GAIN Gain of R channel.

TY_INT_G_GAIN Gain of G channel.

TY_INT_B_GAIN Gain of B channel.

TY_STRUCT_WORK_MODE mode of trigger

Definition at line 188 of file TY API.h.

4.1.2.3 enum TY_IMAGE_MODE_LIST

Enumerator

TY_IMAGE_MODE_320x240 1310960
TY_IMAGE_MODE_640x480 2621920
TY_IMAGE_MODE_1280x960 5243840

TY_IMAGE_MODE_2592x1944 10618776

Definition at line 232 of file TY_API.h.

4.1.3 Function Documentation

4.1.3.1 TY_CAPI TYClearBufferQueue (TY_DEV_HANDLE hDevice)

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

Parameters

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

Return values

TY_STATUS_OK	Succeed.				
TY_STATUS_INVALID_HANDLE	Invalid device handle.				
TY_STATUS_BUSY	Device is capturing.				

4.1.3.2 TY_CAPI TYCloseDevice (TY_DEV_HANDLE hDevice)

Close device by device handle.

Parameters

in	hDevice	Device handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_IDLE	Device has been closed.

4.1.3.3 TY_CAPI TYDeinitLib (void)

Deinit this library.

Return values

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

4.1.3.4 TY_CAPI TYDepthToWorld (TY_DEV_HANDLE hDevice, const TY_VECT_3F * depth, TY_VECT_3F * world, int32_t worldPaddingBytes, int32_t pointCount)

- padding bytes could be 0
- · for PCL, world coordinate padding size should be calculated based on the point type

Parameters

in	hDevice	Device handle.
in	depth	Depth values.
out	world	World coordinate.
in	worldPaddingBytes	Number of world padding bytes.
in	pointCount	Number of points to be calculated.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	pDepth or pWorld is NULL.
TY_STATUS_INVALID_PARAMETER	worldPaddingBytes is not 4x.

4.1.3.5 TY_CAPI TYDisableComponents (TY_DEV_HANDLE hDevice, int32_t componentlDs)

Disable components.

Parameters

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

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

4.1.3.6 TY_CAPI TYEnableComponents (TY_DEV_HANDLE hDevice, int32_t componentlDs)

Enable components.

Parameters

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

Return values

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

4.1.3.7 TY_CAPI TYEnqueueBuffer (TY_DEV_HANDLE hDevice, void * buffer, int32_t bufferSize)

Enqueue a user allocated buffer.

Parameters

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

Return values

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

4.1.3.8 TY_CAPI TYEnterDeveloperMode (TY_DEV_HANDLE hDevice)

Enable developer mode by device handle.

Parameters

in	hDevice	Device handle.

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_DEVICE_ERROR	Enter developer mode failed.

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

Get error information.

Parameters

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

Returns

Error string.

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

Fetch one frame.

Parameters

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

Return values

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

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

Get value of bool feature.

Parameters

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

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.

Return values

TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_BOOL.
TY_STATUS_NULL_POINTER	value is NULL.

4.1.3.12 TY_CAPI TYGetComponentlDs (TY_DEV_HANDLE hDevice, int32_t * componentlDs)

Get all components IDs.

Parameters

in	hDevice	Device handle.
out	componentIDs	All component IDs this device has.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	componentIDs is NULL.

4.1.3.13 TY_CAPI TYGetDeviceInfo (TY_DEV_HANDLE hDevice, TY_DEVICE_BASE_INFO * info)

Get base info of the open device.

Parameters

in	hDevice	Device handle.
out	info	Base info out.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	componentIDs is NULL.

4.1.3.14 TY_CAPI TYGetDeviceList (TY_DEVICE_BASE_INFO * deviceInfos, int32_t bufferCount, int32_t * filledDeviceCount)

Get device info list.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	deviceInfos or filledDeviceCount is NULL.

4.1.3.15 TY_CAPI TYGetDeviceNumber (int32_t * deviceNumber)

Get number of current connected devices.

Parameters

	out	deviceNumber	Number of connected devices.
۱	Out	action tailiboi	realised of confidence acrisco.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	deviceNumber is NULL.

$4.1.3.16 \quad TY_CAPI\ TYGetEnabledComponentIDs\ (\ TY_DEV_HANDLE\ \textit{hDevice},\ int 32_t*componentIDs\)$

Get all enabled components IDs.

Parameters

in	hDevice	Device handle.
out	componentIDs	Enabled component IDs.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	componentIDs is NULL.

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

Return values

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

4.1.3.18 TY_CAPI TYGetEnumEntryCount (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, int32_t * entryCount)

Get number of enum entries.

Parameters

in	hDevice	Device handle.
in	componentID	Component ID.
	(, , , , , , , , , , , , , , , , , , ,	
in	featureID	Feature ID.

Return values

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

4.1.3.19 TY_CAPI TYGetEnumEntryInfo (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, TY_ENUM_ENTRY * entries, int32_t entryCount, int32_t * filledEntryCount)

Get list of enum entries.

Parameters

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

TY_STATUS_OK	Succeed.

Return values

TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_ENUM.
TY_STATUS_NULL_POINTER	entries or filledEntryCount is NULL.

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

Get feature info.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_NULL_POINTER	featureInfo is NULL.

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

Get value of float feature.

Parameters

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

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

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

Get value range of float feature.

Parameters

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

Return values

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

4.1.3.23 TY_CAPI TYGetFrameBufferSize (TY_DEV_HANDLE hDevice, int32_t * bufferSize)

Get total buffer size of one frame in current configuration.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	bufferSize is NULL.

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

Get value of integer feature.

Parameters

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

Return values

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

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

Get value range of integer feature.

Parameters

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

Return values

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

4.1.3.26 TY_CAPI TYGetString (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, char * buffer, int32_t bufferSize)

Get value of string feature.

Parameters

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

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.

Return values

TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_STRING.
TY_STATUS_NULL_POINTER	buffer is NULL.

4.1.3.27 TY_CAPI TYGetStringBufferSize (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, int32_t * size)

Get internal buffer size of string feature.

Parameters

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

Return values

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

4.1.3.28 TY_CAPI TYGetStruct (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, void * pStruct, int32_t structSize)

Get value of struct.

Parameters

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

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.

Return values

TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_STRUCT.
TY_STATUS_NULL_POINTER	pStruct is NULL.
TY_STATUS_WRONG_SIZE	structSize incorrect.

4.1.3.29 TY_CAPI TYIsCapturing (TY_DEV_HANDLE hDevice, bool * isCapturing)

Get if the device is capturing.

Parameters

in	hDevice	Device handle.
out	isCapturing	Return capturing status.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	isCapturing is NULL.

4.1.3.30 TY_CAPI TYLibVersion (TY_VERSION_INFO * version)

Get current library version.

Parameters

out	version	Version infomation to be filled.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NULL_POINTER	buffer is NULL.

4.1.3.31 TY_CAPI TYOpenDevice (const char * deviceID, TY_DEV_HANDLE * deviceHandle)

Open device by device ID.

Parameters

in	deviceID	Device ID string, can be get from TY_DEVICE_BASE_INFO.	
out	deviceHandle	Handle of opened device.	

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	deviceID or deviceHandle is NULL.
TY_STATUS_INVALID_PARAMETER	Device not found.
TY_STATUS_BUSY	Device has been opened.
TY_STATUS_DEVICE_ERROR	Open device failed.

4.1.3.32 TY_CAPI TYOpenDeviceWithIP (const char * IP, TY_DEV_HANDLE * deviceHandle)

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

Parameters

in	IP	Device IP.
out	deviceHandle	Handle of opened device.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	IP or deviceHandle is NULL.
TY_STATUS_INVALID_PARAMETER	Device not found.
TY_STATUS_BUSY	Device has been opened, may occupied somewhere else.
TY_STATUS_DEVICE_ERROR	Open device failed.

4.1.3.33 TY_CAPI TYRegisterCallback (TY_DEV_HANDLE hDevice, TY_FRAME_CALLBACK callback, void * userdata)

Register callback of frame. Register NULL to clean callback.

Parameters

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

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_BUSY	Device is capturing.

 $\textbf{4.1.3.34} \quad \textbf{TY_CAPI TYRegisterEventCallback (TY_DEV_HANDLE \textit{hDevice, TY_EVENT_CALLBACK } \textit{callback, } \textit{void} * \textit{userdata })$

Register device status callback. Register NULL to clean callback.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_BUSY	Device is capturing.

4.1.3.35 TY_CAPI TYSendSoftTrigger (TY_DEV_HANDLE hDevice)

Send a software trigger when device works in trigger mode.

Parameters

in hDevice Device hand

Return values

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

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

Set value of bool feature.

Parameters

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

TY_STATUS_OK Succeed.

Return values

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

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

Return values

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

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

Set value of float feature.

Parameters

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

TY_STATUS_OK	Succeed.

Return values

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

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

Set value of integer feature.

Parameters

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

Return values

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

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

Set value of string feature.

Parameters

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

Return values

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

4.1.3.41 TY_CAPI TYSetStruct (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, void * pStruct, int32_t structSize)

Set value of struct.

Parameters

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

Return values

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

4.1.3.42 TY_CAPI TYStartCapture (TY_DEV_HANDLE hDevice)

Start capture.

Parameters

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

Return values

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

4.1.3.43 TY_CAPI TYStopCapture (TY_DEV_HANDLE hDevice)

Stop capture.

Parameters

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

Return values

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

4.1.3.44 TY_CAPI TYUndistortImage (const TY_CAMERA_INTRINSIC * cameraIntrinsic, const TY_CAMERA_DISTORTION * cameraDistortion, const TY_CAMERA_INTRINSIC * cameraNewIntrinsic, const TY_IMAGE_DATA * srcImage, TY_IMAGE_DATA * dstImage)

Correct image for lens distortion Format of source image data should be TY_PIXEL_FORMAT_MONO or TY_PI \leftarrow XEL_FORMAT_RGB Output buffer is allocated by caller. For IR image undistortion, enable TY_BOOL_UNDIST \leftarrow ORTION to get better performance.

Parameters

in	cameraIntrinsic	input image camera intrinsic parameters
in	cameraDistortion	input image camera distortion parameters
in	cameraNewIntrinsic	output image camera intrinsic , cameraIntrinsic will be used if is NULL
in	srcImage	input image buffer
out	dstImage	Output image buffer

TY_STATUS_OK	Succeed.
TY_STATUS_NULL_POINTER	buffer is NULL.
TY_STATUS_INVALID_PARAMETER	parameter is invalid.
TY_STATUS_ERROR	internal error

4.1.3.45 TY_CAPI TYWorldToDepth (TY_DEV_HANDLE hDevice, const TY_VECT_3F * world, TY_VECT_3F * depth, int32_t worldPaddingBytes, int32_t pointCount)

- padding bytes could be 0
- for PCL, world coordinate padding size should be calculated based on the point type

Parameters

in	hDevice	Device handle.
in	world	World coordinate.
out	depth	Depth values.
in	worldPaddingBytes	Number of depth padding bytes.
in	pointCount	Number of points to be calculated.

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	pDepth or pWorld is NULL.
TY_STATUS_INVALID_PARAMETER	worldPaddingBytes is not 4x.

Index

TY_API.h, 13	TYErrorString, 23
TY_BOOL_AUTO_AWB, 19	TYFetchFrame, 23
TY BOOL AUTO EXPOSURE, 19	TYGetBool, 23
TY_BOOL_AUTO_GAIN, 19	TYGetComponentIDs, 24
TY BOOL BRIGHTNESS HISTOGRAM, 19	TYGetDeviceInfo, 24
TY BOOL LASER AUTO CTRL, 19	TYGetDeviceList, 24
TY BOOL TRIGGER MODE, 19	TYGetDeviceNumber, 25
TY_BOOL_UNDISTORTION, 19	TYGetEnabledComponentIDs, 25
TY COMPONENT BRIGHT HISTO, 19	TYGetEnum, 25
TY_COMPONENT_DEPTH_CAM, 19	TYGetEnumEntryCount, 26
TY COMPONENT DEVICE, 19	TYGetEnumEntryInfo, 26
TY COMPONENT IMU, 19	TYGetFeatureInfo, 27
	TYGetFloat, 27
TY_COMPONENT_IR_CAM_RIGHT_10	
TY_COMPONENT_IR_CAM_RIGHT, 19	TYGetFromePufferSize 28
TY_COMPONENT_LASER, 19	TYGetlet 88
TY_COMPONENT_POINT3D_CAM, 19	TYGetletBarra 00
TY_COMPONENT_RGB_CAM_LEFT, 19	TYGetIntRange, 29
TY_COMPONENT_RGB_CAM_RIGHT, 19	TYGetString, 29
TY_DEVICE_COMPONENT_LIST, 19	TYGetStringBufferSize, 30
TY_ENUM_IMAGE_MODE, 19	TYGetStruct, 30
TY_ENUM_PIXEL_FORMAT, 19	TYIsCapturing, 31
TY_ENUM_TRIGGER_ACTIVATION, 19	TYLibVersion, 31
TY_FEATURE_ID_LIST, 19	TYOpenDevice, 31
TY_IMAGE_MODE_1280x960, 20	TYOpenDeviceWithIP, 32
TY_IMAGE_MODE_2592x1944, 20	TYRegisterCallback, 32
TY_IMAGE_MODE_320x240, 20	TYRegisterEventCallback, 32
TY_IMAGE_MODE_640x480, 20	TYSendSoftTrigger, 33
TY_IMAGE_MODE_LIST, 19	TYSetBool, 33
TY_INT_B_GAIN, 19	TYSetEnum, 34
TY_INT_EXPOSURE_TIME, 19	TYSetFloat, 34
TY_INT_FRAME_PER_TRIGGER, 19	TYSetInt, 35
TY_INT_G_GAIN, 19	TYSetString, 35
TY_INT_GAIN, 19	TYSetStruct, 36
TY_INT_LASER_POWER, 19	TYStartCapture, 36
TY_INT_R_GAIN, 19	TYStopCapture, 37
TY_STRUCT_CAM_DISTORTION, 19	TYUndistortImage, 37
TY_STRUCT_CAM_INTRINSIC, 19	TYWorldToDepth, 37
TY_STRUCT_EXTRINSIC_TO_LEFT_IR, 19	TY_BOOL_AUTO_AWB
TY_STRUCT_EXTRINSIC_TO_LEFT_RGB, 19	TY_API.h, 19
TY_STRUCT_NET_INFO, 19	TY_BOOL_AUTO_EXPOSURE
TY_STRUCT_WORK_MODE, 19	TY_API.h, 19
TYClearBufferQueue, 20	TY_BOOL_AUTO_GAIN
TYCloseDevice, 20	TY_API.h, 19
TYDeinitLib, 20	TY_BOOL_BRIGHTNESS_HISTOGRAM
TYDepthToWorld, 21	TY_API.h, 19
TYDisableComponents, 21	TY_BOOL_LASER_AUTO_CTRL
TYEnableComponents, 22	TY_API.h, 19
TYEnqueueBuffer, 22	TY_BOOL_TRIGGER_MODE
TYEnterDeveloperMode, 22	TY_API.h, 19

40 INDEX

TY_BOOL_UNDISTORTION	TY_API.h, 19
TY_API.h, 19	TY_INT_G_GAIN
TY_CAMERA_DISTORTION, 5	TY_API.h, 19
TY_CAMERA_EXTRINSIC, 5	TY_INT_GAIN
TY_CAMERA_INTRINSIC, 6	TY_API.h, 19
TY_COMPONENT_BRIGHT_HISTO	TY_INT_LASER_POWER
TY API.h, 19	TY API.h, 19
TY_COMPONENT_DEPTH_CAM	TY INT R GAIN
TY API.h, 19	TY API.h, 19
TY_COMPONENT_DEVICE	TY_INT_RANGE, 10
TY API.h, 19	TY_STRUCT_CAM_DISTORTION
-	
TY_COMPONENT_IMU	TY_API.h, 19
TY_API.h, 19	TY_STRUCT_CAM_INTRINSIC
TY_COMPONENT_IR_CAM_LEFT	TY_API.h, 19
TY_API.h, 19	TY_STRUCT_EXTRINSIC_TO_LEFT_IR
TY_COMPONENT_IR_CAM_RIGHT	TY_API.h, 19
TY_API.h, 19	TY_STRUCT_EXTRINSIC_TO_LEFT_RGB
TY_COMPONENT_LASER	TY_API.h, 19
TY_API.h, 19	TY_STRUCT_NET_INFO
TY_COMPONENT_POINT3D_CAM	TY_API.h, 19
	TY_STRUCT_WORK_MODE
TY COMPONENT RGB CAM LEFT	TY API.h, 19
TY_API.h, 19	TY_TRIGGER_MODE, 11
TY_COMPONENT_RGB_CAM_RIGHT	TY_VECT_3F, 11
TY_API.h, 19	TY_VERSION_INFO, 11
TY_DEVICE_BASE_INFO, 6	TYClearBufferQueue
TY_DEVICE_COMPONENT_LIST	TY_API.h, 20
TY_API.h, 19	TYCloseDevice
TY_DEVICE_NET_INFO, 7	TY_API.h, 20
TY_ENUM_ENTRY, 7	TYDeinitLib
TY_ENUM_IMAGE_MODE	TY_API.h, 20
TY_API.h, 19	TYDepthToWorld
TY_ENUM_PIXEL_FORMAT	TY API.h, 21
TY API.h, 19	TYDisableComponents
TY_ENUM_TRIGGER_ACTIVATION	TY_API.h, 21
TY_API.h, 19	TYEnableComponents
	•
TY_EVENT_INFO, 8	TY_API.h, 22
TY_FEATURE_ID_LIST	TYEnqueueBuffer
TY_API.h, 19	TY_API.h, 22
TY_FEATURE_INFO, 8	TYEnterDeveloperMode
TY_FLOAT_RANGE, 9	TY_API.h, 22
TY_FRAME_DATA, 9	TYErrorString
TY_IMAGE_DATA, 10	TY_API.h, 23
TY_IMAGE_MODE_1280x960	TYFetchFrame
TY API.h, 20	TY API.h, 23
TY_IMAGE_MODE_2592x1944	TYGetBool
TY API.h, 20	TY API.h, 23
TY_IMAGE_MODE_320x240	TYGetComponentIDs
TY API.h, 20	TY API.h, 24
-	
TY_IMAGE_MODE_640x480	TYGetDeviceInfo
TY_API.h, 20	TY_API.h, 24
TY_IMAGE_MODE_LIST	TYGetDeviceList
TY_API.h, 19	TY_API.h, 24
TY_INT_B_GAIN	TYGetDeviceNumber
TY_API.h, 19	TY_API.h, 25
TY_INT_EXPOSURE_TIME	TYGetEnabledComponentIDs
TY API.h, 19	TY API.h, 25
TY_INT_FRAME_PER_TRIGGER	TYGetEnum
	

INDEX 41

TY_API.h, 25 TYGetEnumEntryCount TY_API.h, 26 TYGetEnumEntryInfo TY_API.h, 26 TYGetFeatureInfo TY API.h, 27 **TYGetFloat** TY API.h, 27 TYGetFloatRange TY_API.h, 28 TYGetFrameBufferSize TY_API.h, 28 **TYGetInt** TY_API.h, 28 TYGetIntRange TY API.h, 29 **TYGetString** TY_API.h, 29 TYGetStringBufferSize TY_API.h, 30 **TYGetStruct** TY_API.h, 30 **TYIsCapturing** TY_API.h, 31 **TYLibVersion** TY_API.h, 31 **TYOpenDevice** TY API.h, 31 TYOpenDeviceWithIP TY_API.h, 32 TYRegisterCallback TY_API.h, 32 TYRegisterEventCallback TY_API.h, 32 TYSendSoftTrigger TY_API.h, 33 TYSetBool TY_API.h, 33 **TYSetEnum** TY_API.h, 34 TYSetFloat TY_API.h, 34 TYSetInt TY_API.h, 35 **TYSetString** TY_API.h, 35 **TYSetStruct** TY API.h, 36 **TYStartCapture** TY_API.h, 36 **TYStopCapture** TY_API.h, 37 TYUndistortImage TY_API.h, 37 TYWorldToDepth

TY_API.h, 37