Camport2 Project

1.1

Generated by Doxygen 1.8.11

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 TY_CAMERA_DISTORTION Struct Reference

Data Fields

- float **data** [12]

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>
```

Data Fields

- float **data** [4 ∗4]

3.2.1 Detailed Description

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

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>
```

Data Fields

- float **data** [3 ∗3]

3.3.1 Detailed Description

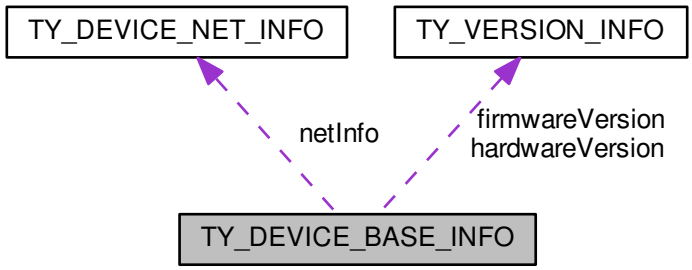
[fx, 0, cx, 0, fy, cy, 0, 0, 1]

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

- TY_API.h

3.4 TY_DEVICE_BASE_INFO Struct Reference

Collaboration diagram for TY_DEVICE_BASE_INFO:



Data Fields

- 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]

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

- TY_API.h

3.5 TY_DEVICE_NET_INFO Struct Reference

Data Fields

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

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

- TY_API.h

3.6 TY_ENUM_ENTRY Struct Reference

Data Fields

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

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

- TY_API.h

3.7 TY_FEATURE_INFO Struct Reference

Data Fields

- 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]

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

- TY_API.h

3.8 TY_FLOAT_RANGE Struct Reference

Data Fields

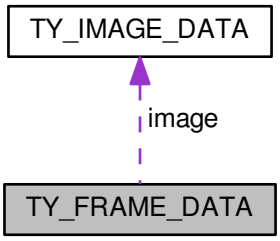
- float **min**
- float **max**
- float **inc**
- float **reserved** [1]

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

- TY_API.h

3.9 TY_FRAME_DATA Struct Reference

Collaboration diagram for TY_FRAME_DATA:



Data Fields

- 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.

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

- TY_API.h

3.10 TY_IMAGE_DATA Struct Reference

Data Fields

- int32_t timestamp

  Timestamp in milliseconds.
- 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.

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

- TY_API.h

3.11 TY_INT_RANGE Struct Reference

Data Fields

- int32_t **min**
- int32_t **max**
- int32_t **inc**
- int32_t **reserved** [1]

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

- TY_API.h

3.12 TY_VECT_3F Struct Reference

Data Fields

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

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

- TY_API.h

3.13 TY_VERSION_INFO Struct Reference

Data Fields

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

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

- TY_API.h

Chapter 4

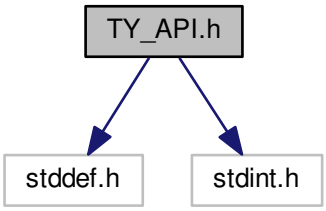
File Documentation

4.1 TY_API.h File Reference

API for Percipio depth cameras.

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

Include dependency graph for TY_API.h:



Data Structures

- 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
- struct TY_IMAGE_DATA
- struct TY_FRAME_DATA

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** 2
- #define **TY_LIB_VERSION_PATCH** 0
- #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 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 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_IMAGE_DATA **TY_IMAGE_DATA**
- typedef struct TY_FRAME_DATA **TY_FRAME_DATA**
- typedef void(* **TY_FRAME_CALLBACK**) (TY_FRAME_DATA *, 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_INVALID_PARAMETER** = -1006, **TY_STATUS_INVALID_HANDLE** = -1007,
  TY_STATUS_INVALID_COMPONENT = -1008, **TY_STATUS_INVALID_FEATURE** = -1009, **TY_STATUS_WRONG_TYPE** = -1010, **TY_STATUS_WRONG_SIZE** = -1011,
  TY_STATUS_OUT_OF_MEMORY = -1012, **TY_STATUS_OUT_OF_RANGE** = -1013, **TY_STATUS_TIMEOUT** = -1014, **TY_STATUS_WRONG_MODE** = -1015,
  TY_STATUS_BUSY = -1016, **TY_STATUS_IDLE** = -1017, **TY_STATUS_NO_DATA** = -1018, **TY_STATUS_NULL_POINTER** = -1020 }
- enum TY_DEVICE_COMPONENT_LIST {
  TY_COMPONENT_DEVICE = 0x80000000, TY_COMPONENT_DEPTH_CAM = 0x00010000, TY_COMPONENT_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_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 }
- enum TY_FEATURE_ID_LIST {
  TY_STRUCT_CAM_INTRINSIC = 0x000 | TY_FEATURE_STRUCT, TY_STRUCT_EXTRINSIC_TO_LEFT_IR = 0x001 | TY_FEATURE_STRUCT, TY_STRUCT_EXTRINSIC_TO_LEFT_RGB = 0x002 | TY_FEATURE_STRUCT, TY_STRUCT_NET_INFO = 0x005 | TY_FEATURE_STRUCT,
  TY_STRUCT_CAM_DISTORTION = 0x006 | TY_FEATURE_STRUCT, **TY_INT_WIDTH_MAX** = 0x100 | TY_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_INT_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_TRIGGER_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_INT_LASER_POWER = 0x500 | TY_FEATURE_INT, TY_BOOL_UNDISTORTION = 0x510 | 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 }
- enum **TY_IMAGE_MODE_LIST** { **TY_IMAGE_MODE_320x240** = (320<<12)+240, **TY_IMAGE_MODE_640x480** = (640<<12)+480, **TY_IMAGE_MODE_1280x960** = (1280<<12)+960 }
- enum **TY_TRIGGER_ACTIVATION_LIST** { **TY_TRIGGER_ACTIVATION_FALLINGEDGE** = 0, **TY_TRIGGER_ACTIVATION_RISINGEDGE** = 1 }
- enum **TY_INTERFACE_LIST** { **TY_INTERFACE_UNKNOWN** = 0, **TY_INTERFACE_ETHERNET** = 1, **TY_INTERFACE_USB** = 2 }
- enum **TY_ACCESS_MODE_LIST** { **TY_ACCESS_READABLE** = 0x1, **TY_ACCESS_WRITABLE** = 0x2 }
- enum **TY_PIXEL_TYPE_LIST** { **TY_PIXEL_MONO** = 0x10000000, **TY_PIXEL_COLOR** = 0x20000000, **TY_PIXEL_DEPTH** = 0x30000000, **TY_PIXEL_POINT3D** = 0x40000000 }
- enum **TY_PIXEL_BITS_LIST** {
  TY_PIXEL_8BIT = 0x00080000, **TY_PIXEL_16BIT** = 0x00100000, **TY_PIXEL_24BIT** = 0x00180000, **TY_PIXEL_32BIT** = 0x00200000,
  TY_PIXEL_96BIT = 0x00600000 }

- enum **TY_PIXEL_FORMAT_LIST** {
  TY_PIXEL_FORMAT_UNDEFINED = 0, **TY_PIXEL_FORMAT_MONO** = (TY_PIXEL_MONO | TY_PIXEL↩
  _8BIT | 0x0000), **TY_PIXEL_FORMAT_RGB** = (TY_PIXEL_COLOR | TY_PIXEL_24BIT | 0x0010), **TY_PI↩
  XEL_FORMAT_YUV422** = (TY_PIXEL_COLOR | TY_PIXEL_16BIT | 0x0011),
  TY_PIXEL_FORMAT_YUYV = (TY_PIXEL_COLOR | TY_PIXEL_16BIT | 0x0012), **TY_PIXEL_FORMAT↩
  _DEPTH16** = (TY_PIXEL_DEPTH | TY_PIXEL_16BIT | 0x0020), **TY_PIXEL_FORMAT_FPOINT3D** = (TY↩
  _PIXEL_POINT3D | TY_PIXEL_96BIT | 0x0030) }

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 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 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_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, int32_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, 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)

  Convert points on depth image to world coordinate. format of depth data should be: +------+------+------+------+------+------+---- | p0.x | p0.y | p0.z | p1.x | p1.y | p1.z | ... +------+------+------+------+------+------+---- and world coordinate should be: +------+------+------+--------------+------+------+------+--------------+---- | p0.x | p0.y | p0.z | padding bytes | p1.x | p1.y | p1.z | padding bytes | ... +------+------+------+--------------+------+------+------+--------------+----.

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

  Convert world coordinate to depth coordinate. format of depth data should be: +------+------+------+------+------+------+---- | p0.x | p0.y | p0.z | p1.x | p1.y | p1.z | ... +------+------+------+------+------+------+---- and world coordinate should be: +------+------+------+--------------+------+------+------+--------------+---- | p0.x | p0.y | p0.z | padding bytes | p1.x | p1.y | p1.z | padding bytes | ... +------+------+------+--------------+------+------+------+--------------+----.

- TY_CAPI **_TYInitLib** (void)
- TY_CAPI **TYGetEnabledComponents** (TY_DEV_HANDLE hDevice, int32_t *componentIDs)
- 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.

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.

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_CAMERA_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 microseconds.

TY_BOOL_AUTO_GAIN Auto gain switch.

TY_INT_GAIN Gain.

TY_INT_LASER_POWER Laser power level.

TY_BOOL_UNDISTORTION Output undistorted image.

TY_INT_R_GAIN Gain of R channel.

TY_INT_G_GAIN Gain of G channel.

TY_INT_B_GAIN Gain of B channel.

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*)

Convert points on depth image to world coordinate. format of depth data should be: +---—+---—+---—+---—+---—+---—+—- | p0.x | p0.y | p0.z | p1.x | p1.y | p1.z | ... +---—+---—+---—+---—+---—+---—+---—+—- and world coordinate should be: +---—+---—+---—+----------—+---—+---—+---—+----------—+—- | p0.x | p0.y | p0.z | padding bytes | p1.x | p1.y | p1.z | padding bytes | ... +---—+---—+---—+----------—+---—+---—+---—+----------—+—.

- 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 *componentIDs*)

Disable components.

Parameters

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

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.

4.1.3.6 TY_CAPI TYEnableComponents (TY_DEV_HANDLE *hDevice,* int32_t *componentIDs*)

Enable components.

Parameters

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.

Return values

TY_STATUS_NULL_POINTER	buffer is NULL.
TY_STATUS_WRONG_SIZE	The input buffer is not large enough.

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

Get error information.

Parameters

in	*errorID*	Error id.

Returns

Error string.

4.1.3.9 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.

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

Get value of bool feature.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	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_BOOL.
TY_STATUS_NULL_POINTER	value is NULL.

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

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.

4.1.3.12 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.

4.1.3.13 TY_CAPI TYGetDeviceNumber (int32_t * *deviceNumber*)

Get number of current connected devices.

Parameters

out	*deviceNumber*	Number of connected devices.

Return values

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

4.1.3.14 TY_CAPI TYGetEnabledComponentIDs (TY_DEV_HANDLE *hDevice*, int32_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.

4.1.3.15 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.

4.1.3.16 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.
out	*entryCount*	Entry count.

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.

4.1.3.17 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.

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	entries or filledEntryCount is NULL.

4.1.3.18 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.

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

Get value of float feature.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	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.

4.1.3.20 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.

Parameters

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.

4.1.3.21 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.

4.1.3.22 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.

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_INT.
TY_STATUS_NULL_POINTER	value is NULL.

4.1.3.23 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.

4.1.3.24 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.

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.

Return values

TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_STRING.
TY_STATUS_NULL_POINTER	buffer is NULL.

4.1.3.25 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.

4.1.3.26 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..

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_STRUCT.
TY_STATUS_NULL_POINTER	pStruct is NULL.

Return values

TY_STATUS_WRONG_SIZE	structSize incorrect.

4.1.3.27 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.

4.1.3.28 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.

4.1.3.29 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.

Return values

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.

4.1.3.30 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.

4.1.3.31 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.

Return values

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

4.1.3.32 TY_CAPI TYSendSoftTrigger (TY_DEV_HANDLE *hDevice*)

Send a software trigger when device works in trigger mode.

Parameters

in	*hDevice*	Device handle.

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.

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

Set value of bool feature.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	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.

4.1.3.34 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.

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

Set value of float feature.

Parameters

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

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	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.

4.1.3.36 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.

4.1.3.37 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.

4.1.3.38 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.

4.1.3.39 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.

4.1.3.40 TY_CAPI TYStopCapture (TY_DEV_HANDLE *hDevice*)

Stop capture.

Parameters

in	*hDevice*	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.

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

Convert world coordinate to depth coordinate. format of depth data should be: +---—+---—+---—+---—+---—+---—+—- | p0.x | p0.y | p0.z | p1.x | p1.y | p1.z | ... +---—+---—+---—+---—+---—+---—+—- and world coordinate should be: +---—+---—+---—+----------—+---—+---—+---—+----------—+—- | p0.x | p0.y | p0.z | padding bytes | p1.x | p1.y | p1.z | padding bytes | ... +---—+---—+---—+----------—+---—+---—+---—+----------—+—-.

- 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.

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.

Index