MYNT EYE S SDK

2.3.3

Generated by Doxygen 1.8.15

Chapter 1

MYNT EYE S SDK

- API Classes
- API Modules
  - Enumerations
  - Datatypes
  - Utiliities
  - Intrinsics & Extrinsics
- Device Specifications
  - Device Data Specification
  - Device Control Specification

Chapter 2

Device Data Specification

2.1 Device Information

Name	Field	Fixed Value	USB Descriptor	UVC Extension Unit	Bytes	Note
VID	vid	0x04B4	✓	×	2	
PID	pid	0x00F9	✓	×	2	
Device name	name	MYNT-EYE-?	✓	✓ Get	16	MYNT-EYE-S1000
Serial number	serial_↩ number	-	✓	✓ Get	16	
Firmware version	firmware_↩ version	-	✓	✓ Get	2	major,minor
Hardware version	hardware_↩ version	-	×	✓ Get	3	major,minor,flag
Spec version	spec_version	-	×	✓ Get	2	major,minor
Lens type	lens_type	-	×	✓ Get/Set	4	vendor(2),product(2); default: 0
IMU type	imu_type	-	×	✓ Get/Set	4	vendor(2),product(2); default: 0
Nominal baseline	nominal_↩ baseline	-	×	✓ Get/Set	2	unit: mm; default: 0
Auxiliary chip version	auxiliary_↩ chip_version	-	×	✓ Get	2	major,minor
isp version	isp_version	-	×	✓ Get	2	major,minor

2.2 Image Params

Image Intrinsics

Name	Field	Unit	Bytes	Note
Image width	width	px	2	uint16_t; [0,65535]
Image height	height	px	2	uint16_t; [0,65535]
Focal length	fx	-	8	double
	fy	-	8	double
Principal point	cx	-	8	double
	cy	-	8	double
Distortion model	model	-	1	uint8_t; pinhole,...
Distortion coefficients	coeffs[5]	-	40	double; k1,k2,p1,p2,k3

Image Extrinsics

Transformation matrix from left image to right image.

Name	Field	Unit	Bytes	Note
Rotation matrix	rotation[3][3]	-	72	double
Translation vector	translation[3]	-	24	double

2.3 IMU Params

IMU Intrinsics

Name	Field	Unit	Bytes	Note
Scale matrix	acc_scale[3][3]	-	72	double
	gyro_scale[3][3]	-	72	double
Zero-drift	acc_drift[3]	-	24	double
	gyro_drift[3]	-	24	double
Noise density	acc_noise[3]	-	24	double
	gyro_noise[3]	-	24	double
Random walk	acc_bias[3]	-	24	double
	gyro_bias[3]	-	24	double

IMU Extrinsics

Transformation matrix from left image to IMU.

Name	Field	Unit	Bytes	Note
Rotation matrix	rotation[3][3]	-	72	double
Translation vector	translation[3]	-	24	double

2.4 Image Data

Name	Field	Unit	Bytes	Note
Frame ID	frame_id	-	2	uint16_t; [0,65535]
Timestamp	timestamp	10 us	4	uint32_t
Exposure Time	exposure_time	10 us	2	uint16_t

Image Packet

Name	Header	Size	Frame ID	Timestamp	Exposure Time	Checksum
Bytes	1	1	2	4	2	1
Type	uint8_t	uint8_t	uint16_t	uint32_t	uint16_t	uint8_t
Description	0x3B	0x08, content size	Frame ID	Timestamp	Exposure time	Checksum, XOR of all content bytes

- The image packet will be dropped, if checksum is incorrect.
- The accuracy of the time unit: 0.01 ms / 10 us.
  - The timestamp could indicate 11.9 hours, it will accumulate again after overflow.
- The timestamp accumulation starts from the time of power-on, instead of opening.

2.5 IMU Data

IMU Request Packet

Name	Header	Serial Number
Bytes	1	4
Type	uint8_t	uint32_t
Description	0x5A	First request should be 0, otherwise the last one

IMU Response Packet

The IMU response packet contains multiple IMU packets, and each IMU packet contains multiple IMU segments.

Name	Header	State	Size	IMU Packets	Checksum
Bytes	1	1	2	...	1
Type	uint8_t	uint8_t	uint16_t	-	uint8_t
Description	0x5B	0 is success, others are failed	Content size	IMU packets	Checksum, XOR of all content bytes

IMU Packet

The IMU packet is an array of IMU datas.

Name	Serial Number	Timestamp	Count	IMU Datas
Bytes	4	4	1	...
Type	uint32_t	uint32_t	uint8_t	-
Description	Serial number	IMU basic timestamp	The number of IMU datas	IMU datas

IMU Segment

Name	Offset	Frame ID	Accelerometer	Temperature	Gyroscope
Bytes	2	2	6	2	6
Type	int16_t	uint16_t	int16_t $*$ 3	int16_t	int16_t $*$ 3
Description	The timestamp offset	Image frame ID	Accel x,y,z values	IMU temperature	Gyro x,y,z values

- Formula for converting the accel & gyro values to real ones: **real = data** $*$ **range** / **0x10000** .
  - `accel` default `range` is **8 g**, `gyro` default `range` is **1000 deg/s**.
- Formula for converting the temperature to real value: **real = data** / **ratio + offset** .
  - default `ratio` is **326.8**, default `offset` is **25℃**.

Chapter 3

Device Control Specification

3.1 Control Protocols

There are two control modes, one is through UVC standard protocol, the other is through UVC custom protocol with extension unit.

Standard Protocol

Name	Field	Bytes	Default	Min	Max	Stored	Flash Address	Note
Gain	gain	2	24	0	48	✓	0x12	valid if manual-exposure
Brightness	brightness/exposure_time	2	120	0	240	✓	0x14	valid if manual-exposure
Contrast	contrast/black_level_calibration	2	127	0	255	✓	0x10	valid if manual-exposure

Custom Protocol

Name	Field	Bytes	Default	Min	Max	Stored	Flash Address	Channel	Note
Frame rate	frame_rate	2	25	10	60	✓	0x21	XU_CAM_CTRL	values: {10,15,20,25,30,35,40,45,50,5
IMU frequency	imu_frequency	2	200	100	500	✓	0x23	XU_CAM_CTRL	values: {100,200,250,333,500}

Name	Field	Bytes	Default	Min	Max	Stored	Flash Address	Channel	Note
Exposure mode	exposure_mode	1	0	0	1	✓	0x0F	XU_CAM_CTRL	0: enable auto-exposure; 1: manual-exposure
Max gain	max_gain	2	48	0	48	✓	0x1D	XU_CAM_CTRL	valid if auto-exposure
Max exposure time	max_exposure_time	2	240	0	240	✓	0x1B	XU_CAM_CTRL	valid if auto-exposure
Desired brightness	desired_brightness	2	192	0	255	✓	0x19	XU_CAM_CTRL	valid if auto-exposure
IR control	ir_control	1	0	0	160	×	-	XU_CAM_CTRL	
HDR mode	hdr_mode	1	0	0	1	✓	0x1F	XU_CAM_CTRL	0: 10-bit; 1: 12-bit
Zero drift calibration	zero_drift_calibration		-	-	-	×	-	XU_HALF_DUPLEX	
Erase chip	erase_chip		-	-	-	×	-	XU_HALF_DUPLEX	

3.2 Control Channels

Name	Field	Address	Bandwidth	Node
Camera control channel	XU_CAM_CTRL_CHANNEL	1	3	
Half-Duplex channel	XU_HALF_DUPLEX_CHANNEL	2	20	
IMU write channel	XU_IMUDATA_WRITE_CHANNEL	3	5	
IMU read channel	XU_IMUDATA_READ_CHANNEL	4	2000	
File channel	XU_FILE_CHANNEL	5	2000	

Camera Control Channel

The channel provides get, set and query (min, max, default).

Half-Duplex Channel

The channel only provides set, such as zero drift correction.

IMU Channel

The channel is used to request and response IMU data, see IMU Data.

File Channel

The channel is used to read and write device information, image params, and IMU params.

Name	Header	Size	File	Checksum
Bytes	1	2	-	1
Type	uint8_t	uint16_t	-	uint8_t
Description	Flags	Content size	Content data	Checksum, XOR of all content bytes

Header Bit Subscript	Description
0	Device information
1	Image params
2	IMU params
3∼6	Undefined
7	0: Get; 1: Set

File Content Packet

Name	ID	Size	Content
Bytes	1	2	-
Type	uint8_t	uint16_t	-
Description	Content ID	Content size	Content data

File	ID	Max Size
Device information	1	250
Image params	2	250
IMU params	4	500

Chapter 4

Deprecated List

Member mynteye::API::GetIntrinsics (const Stream &stream) const

Get the intrinsics (pinhole) of stream.

Member mynteye::Device::GetLatestStreamData (const Stream &stream)

Replaced by GetStreamData(const Stream &stream)

Member mynteye::IntrinsicsPinhole::model

Replaced by calib_model_.

Chapter 5

Module Index

5.1 Modules

Here is a list of all modules:

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 7

Class Index

7.1 Class List

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

Chapter 8

Module Documentation

8.1 Utiliities

Functions

- MYNTEYE_API std::shared_ptr< Device > mynteye::device::select ()

  Detecting MYNT EYE devices and prompt user to select one.
- MYNTEYE_API MYNTEYE_NAMESPACE::StreamRequest mynteye::device::select_request (const std::shared_ptr< Device > &device, bool *ok)

  List stream requests and prompt user to select one.
- MYNTEYE_API float mynteye::utils::get_real_exposure_time (std::int32_t frame_rate, std::uint16_t exposure_time)

  Get real exposure time in ms from virtual value, according to its frame rate.
- MYNTEYE_API std::string mynteye::utils::get_sdk_root_dir ()

  Get sdk root dir.
- MYNTEYE_API std::string mynteye::utils::get_sdk_install_dir ()

  Get sdk install dir.

8.1.1 Detailed Description

8.1.2 Function Documentation

8.1.2.1 get_real_exposure_time()

```
MYNTEYE_API float mynteye::utils::get_real_exposure_time (
            std::int32_t frame_rate,
            std::uint16_t exposure_time )
```

Get real exposure time in ms from virtual value, according to its frame rate.

Parameters

frame_rate	the frame rate of the device.
exposure_time	the virtual exposure time.

Returns

the real exposure time in ms, or the virtual value if frame rate is invalid.

8.1.2.2 select()

```
MYNTEYE_API std::shared_ptr<Device> mynteye::device::select ( )
```

Detecting MYNT EYE devices and prompt user to select one.

Returns

the selected device, or `nullptr` if none.

8.1.2.3 select_request()

```
MYNTEYE_API MYNTEYE_NAMESPACE::StreamRequest mynteye::device::select_request (
            const std::shared_ptr< Device > & device,
            bool * ok )
```

List stream requests and prompt user to select one.

Returns

the selected request.

8.2 Enumerations

Public enumeration types.

Enumerations

- enum mynteye::Model : std::uint8_t { mynteye::Model::STANDARD, mynteye::Model::STANDARD2, mynteye::Model::STANDARD210A, mynteye::Model::LAST }

  Device model.
- enum mynteye::Stream : std::uint8_t { mynteye::Stream::LEFT, mynteye::Stream::RIGHT, mynteye::Stream::LEFT_RECTIFIED, mynteye::Stream::RIGHT_RECTIFIE mynteye::Stream::DISPARITY, mynteye::Stream::DISPARITY_NORMALIZED, mynteye::Stream::DEPTH, mynteye::Stream::POINTS, mynteye::Stream::LAST }

  Streams define different type of data.
- enum mynteye::Capabilities : std::uint8_t { mynteye::Capabilities::STEREO, mynteye::Capabilities::STEREO_COLOR, mynteye::Capabilities::COLOR, mynteye::Capabilities::DEPTH, mynteye::Capabilities::POINTS, mynteye::Capabilities::FISHEYE, mynteye::Capabilities::INFRARED, mynteye::Capabilities::INFRARED2, mynteye::Capabilities::IMU, mynteye::Capabilities::LAST }

  Capabilities define the full set of functionality that the device might provide.
- enum mynteye::Info : std::uint8_t { mynteye::Info::DEVICE_NAME, mynteye::Info::SERIAL_NUMBER, mynteye::Info::FIRMWARE_VERSION, mynteye::Info::HARDWARE_VERSION, mynteye::Info::SPEC_VERSION, mynteye::Info::LENS_TYPE, mynteye::Info::IMU_TYPE, mynteye::Info::NOMINAL_BASELINE mynteye::Info::LAST }

  Camera info fields are read-only strings that can be queried from the device.
- enum mynteye::Option : std::uint8_t { mynteye::Option::GAIN, mynteye::Option::BRIGHTNESS, mynteye::Option::CONTRAST, mynteye::Option::FRAME_RATE, mynteye::Option::IMU_FREQUENCY, mynteye::Option::EXPOSURE_MODE, mynteye::Option::MAX_GAIN, mynteye::Option::MAX_EXPOSURE_TIME, mynteye::Option::MIN_EXPOSURE_TIME, mynteye::Option::DESIRED_BRIGHTNESS, mynteye::Option::IR_CONTROL, mynteye::Option::HDR_MODE, mynteye::Option::ACCELEROMETER_RANGE, mynteye::Option::GYROSCOPE_RANGE, mynteye::Option::ACCELEROMET mynteye::Option::GYROSCOPE_LOW_PASS_FILTER, mynteye::Option::ZERO_DRIFT_CALIBRATION, mynteye::Option::ERASE_CHIP, mynteye::Option::LAST }

  Camera control options define general configuration controls.
- enum mynteye::Source : std::uint8_t { mynteye::Source::VIDEO_STREAMING, mynteye::Source::MOTION_TRACKING, mynteye::Source::ALL, mynteye::Source::LAST }

  Source allows the user to choose which data to be captured.
- enum mynteye::AddOns : std::uint8_t { mynteye::AddOns::INFRARED, mynteye::AddOns::INFRARED2, mynteye::AddOns::LAST }

  Add-Ons are peripheral modules of our hardware.
- enum mynteye::Format : std::uint32_t { mynteye::Format::GREY = ((std::uint32_t)('G') | ((std::uint32_t)('R') << 8) | ((std::uint32_t)('E') << 16) | ((std::uint32_t)('Y') << 24)), mynteye::Format::YUYV = ((std::uint32_t)('Y') | ((std::uint32_t)('U') << 8) | ((std::uint32_t)('Y') << 16) | ((std::uint32_t)('V') << 24)), mynteye::Format::BGR888 = ((std↩ ::uint32_t)('B') | ((std::uint32_t)('G') << 8) | ((std::uint32_t)('R') << 16) | ((std::uint32_t)('3') << 24)), mynteye::Format::RGB888 = ((std::uint32_t)('R') | ((std::uint32_t)('G') << 8) | ((std::uint32_t)('B') << 16) | ((std::uint32_t)('3') << 24)), mynteye::Format::LAST }

  Formats define how each stream can be encoded.

8.2.1 Detailed Description

Public enumeration types.

8.2.2 Enumeration Type Documentation

8.2.2.1 AddOns

`enum mynteye::AddOns : std::uint8_t [strong]`

Add-Ons are peripheral modules of our hardware.

Enumerator

INFRARED	Infrared.
INFRARED2	Second infrared.
LAST	Last guard.

8.2.2.2 Capabilities

`enum mynteye::Capabilities : std::uint8_t [strong]`

Capabilities define the full set of functionality that the device might provide.

Enumerator

STEREO	Provides stereo stream.
STEREO_COLOR	Provide stereo color stream.
COLOR	Provides color stream.
DEPTH	Provides depth stream.
POINTS	Provides point cloud stream.
FISHEYE	Provides fisheye stream.
INFRARED	Provides infrared stream.
INFRARED2	Provides second infrared stream.
IMU	Provides IMU (accelerometer, gyroscope) data.
LAST	Last guard.

8.2.2.3 Format

`enum mynteye::Format : std::uint32_t [strong]`

Formats define how each stream can be encoded.

Enumerator

GREY	Greyscale, 8 bits per pixel.
YUYV	YUV 4:2:2, 16 bits per pixel.
BGR888	BGR 8:8:8, 24 bits per pixel.
RGB888	RGB 8:8:8, 24 bits per pixel.
LAST	Last guard.

8.2.2.4 Info

`enum mynteye::Info : std::uint8_t [strong]`

Camera info fields are read-only strings that can be queried from the device.

Enumerator

DEVICE_NAME	Device name.
SERIAL_NUMBER	Serial number.
FIRMWARE_VERSION	Firmware version.
HARDWARE_VERSION	Hardware version.
SPEC_VERSION	Spec version.
LENS_TYPE	Lens type.
IMU_TYPE	IMU type.
NOMINAL_BASELINE	Nominal baseline.
LAST	Last guard.

8.2.2.5 Model

`enum mynteye::Model : std::uint8_t [strong]`

Device model.

Enumerator

STANDARD	Standard.
STANDARD2	Standard 2.
STANDARD210A	Standard 210a.
LAST	Last guard.

8.2.2.6 Option

`enum mynteye::Option : std::uint8_t [strong]`

Camera control options define general configuration controls.

Enumerator

GAIN	Image gain, valid if manual-exposure. range: [0,48], default: 24
BRIGHTNESS	Image brightness, valid if manual-exposure. range: [0,240], default: 120
CONTRAST	Image contrast, valid if manual-exposure. range: [0,255], default: 127
FRAME_RATE	Image frame rate, must set IMU_FREQUENCY together. values: {10,15,20,25,30,35,40,45,50,55,60}, default: 25
IMU_FREQUENCY	IMU frequency, must set FRAME_RATE together. values: {100,200,250,333,500}, default: 200
EXPOSURE_MODE	Exposure mode. 0: enable auto-exposure 1: disable auto-exposure (manual-exposure)
MAX_GAIN	Max gain, valid if auto-exposure. range of standard 1: [0,48], default: 48 range of standard 2: [0,255], default: 8
MAX_EXPOSURE_TIME	Max exposure time, valid if auto-exposure. range of standard 1: [0,240], default: 240 range of standard 2: [0,1000], default: 333
MIN_EXPOSURE_TIME	min exposure time, valid if auto-exposure range: [0,1000], default: 0
DESIRED_BRIGHTNESS	Desired brightness, valid if auto-exposure. range of standard 1: [0,255], default: 192 range of standard 2: [1,255], default: 122
IR_CONTROL	IR control. range: [0,160], default: 0
HDR_MODE	HDR mode. 0: 10-bit 1: 12-bit
ACCELEROMETER_RANGE	The range of accelerometer. value of standard 1: {4,8,16,32}, default: 8 value of standard 2: {6,12,24,48}, default: 12
GYROSCOPE_RANGE	The range of gyroscope. value of standard 1: {500,1000,2000,4000}, default: 1000 value of standard 2: {250,500,1000,2000,4000}, default: 1000
ACCELEROMETER_LOW_PASS_FILTER	The parameter of accelerometer low pass filter. values: {0,1,2}, default: 2
GYROSCOPE_LOW_PASS_FILTER	The parameter of gyroscope low pass filter. values: {23,64}, default: 64
ZERO_DRIFT_CALIBRATION	Zero drift calibration.
ERASE_CHIP	Erase chip.
LAST	Last guard.

8.2.2.7 Source

```
enum mynteye::Source :  std::uint8_t  [strong]
```

Source allows the user to choose which data to be captured.

Enumerator

VIDEO_STREAMING	Video streaming of stereo, color, depth, etc.
MOTION_TRACKING	Motion tracking of IMU (accelerometer, gyroscope)
ALL	Enable everything together.
LAST	Last guard.

8.2.2.8 Stream

```
enum mynteye::Stream :  std::uint8_t  [strong]
```

Streams define different type of data.

Enumerator

LEFT	Left stream.
RIGHT	Right stream.
LEFT_RECTIFIED	Left stream, rectified.
RIGHT_RECTIFIED	Right stream, rectified.
DISPARITY	Disparity stream.
DISPARITY_NORMALIZED	Disparity stream, normalized.
DEPTH	Depth stream.
POINTS	Point cloud stream.
LAST	Last guard.

8.3 Intrinsics & Extrinsics

Intrinsic and extrinsic properties.

Classes

- struct mynteye::IntrinsicsPinhole

  Stream intrinsics (Pinhole)
- struct mynteye::IntrinsicsEquidistant

  Stream intrinsics (Equidistant: KANNALA_BRANDT)
- struct mynteye::ImuIntrinsics

  IMU intrinsics: scale, drift and variances.
- struct mynteye::MotionIntrinsics

  Motion intrinsics, including accelerometer and gyroscope.
- struct mynteye::Extrinsics

  Extrinsics, represent how the different datas are connected.

Enumerations

- enum mynteye::CalibrationModel : std::uint8_t { mynteye::CalibrationModel::PINHOLE = 0, mynteye::CalibrationModel::KANNAL = 1, mynteye::CalibrationModel::UNKNOW }

  Camera calibration model.

8.3.1 Detailed Description

Intrinsic and extrinsic properties.

8.3.2 Enumeration Type Documentation

8.3.2.1 CalibrationModel

```
enum mynteye::CalibrationModel :  std::uint8_t  [strong]
```

Camera calibration model.

Enumerator

PINHOLE	Pinhole.
KANNALA_BRANDT	Equidistant: KANNALA_BRANDT.
UNKNOW	Unknow.

8.4 Datatypes

Public data types.

Classes

- struct mynteye::api::StreamData

  API stream data.
- struct mynteye::api::MotionData

  API motion data.
- class mynteye::device::Frame

  Frame with raw data.
- struct mynteye::device::StreamData

  Device stream data.
- struct mynteye::device::MotionData

  Device motion data.
- struct mynteye::DeviceInfo

  Device infomation.
- struct mynteye::ImgData

  Image data.
- struct mynteye::ImuData

  IMU data.
- struct mynteye::OptionInfo

  Option info.

8.4.1 Detailed Description

Public data types.

Chapter 9

Class Documentation

9.1 mynteye::API Class Reference

The API class to communicate with MYNT® EYE device.

Public Types

- using stream_callback_t = std::function< void(const api::StreamData &data)>

  The api::StreamData callback.
- using motion_callback_t = std::function< void(const api::MotionData &data)>

  The api::MotionData callback.

Public Member Functions

- Model GetModel () const

  Get the model.
- bool Supports (const Stream &stream) const

  Supports the stream or not.
- bool Supports (const Capabilities &capability) const

  Supports the capability or not.
- bool Supports (const Option &option) const

  Supports the option or not.
- bool Supports (const AddOns &addon) const

  Supports the addon or not.
- StreamRequest SelectStreamRequest (bool *ok) const

  Log all stream requests and prompt user to select one.
- const std::vector< StreamRequest > & GetStreamRequests (const Capabilities &capability) const

  Get all stream requests of the capability.
- void ConfigStreamRequest (const Capabilities &capability, const StreamRequest &request)

  Config the stream request to the capability.
- const StreamRequest & GetStreamRequest (const Capabilities &capability) const

  Get the config stream requests of the capability.
- const std::vector< StreamRequest > & GetStreamRequests () const

  Get all stream requests of the key stream capability.

- void ConfigStreamRequest (const StreamRequest &request)

  Config the stream request to the key stream capability.
- const StreamRequest & GetStreamRequest () const

  Get the config stream requests of the key stream capability.
- std::shared_ptr< DeviceInfo > GetInfo () const

  Get the device info.
- std::string GetInfo (const Info &info) const

  Get the device info.
- IntrinsicsPinhole GetIntrinsics (const Stream &stream) const
- template<typename T >
  T GetIntrinsics (const Stream &stream) const

  Get the intrinsics of stream.
- std::shared_ptr< IntrinsicsBase > GetIntrinsicsBase (const Stream &stream) const

  Get the intrinsics base of stream.
- Extrinsics GetExtrinsics (const Stream &from, const Stream &to) const

  Get the extrinsics from one stream to another.
- MotionIntrinsics GetMotionIntrinsics () const

  Get the intrinsics of motion.
- Extrinsics GetMotionExtrinsics (const Stream &from) const

  Get the extrinsics from one stream to motion.
- void LogOptionInfos () const

  Log all option infos.
- OptionInfo GetOptionInfo (const Option &option) const

  Get the option info.
- std::int32_t GetOptionValue (const Option &option) const

  Get the option value.
- void SetOptionValue (const Option &option, std::int32_t value)

  Set the option value.
- bool RunOptionAction (const Option &option) const

  Run the option action.
- void SetStreamCallback (const Stream &stream, stream_callback_t callback)

  Set the callback of stream.
- void SetMotionCallback (motion_callback_t callback)

  Set the callback of motion.
- bool HasStreamCallback (const Stream &stream) const

  Has the callback of stream.
- bool HasMotionCallback () const

  Has the callback of motion.
- void Start (const Source &source)

  Start capturing the source.
- void Stop (const Source &source)

  Stop capturing the source.
- void WaitForStreams ()

  Wait the streams are ready.
- void EnableStreamData (const Stream &stream)

  Enable the data of stream.
- void DisableStreamData (const Stream &stream)

  Disable the data of stream.
- api::StreamData GetStreamData (const Stream &stream)

  Get the latest data of stream.
- std::vector< api::StreamData > GetStreamDatas (const Stream &stream)

Get the datas of stream.

- void EnableMotionDatas (std::size_t max_size=std::numeric_limits< std::size_t >::max())

  Enable cache motion datas.
- std::vector< api::MotionData > GetMotionDatas ()

  Get the motion datas.
- void EnablePlugin (const std::string &path)

  Enable the plugin.

Static Public Member Functions

- static std::shared_ptr< API > Create (int argc, char *argv[])

  Create the API instance.
- static std::shared_ptr< API > Create (int argc, char *argv[], const std::shared_ptr< Device > &device)

  Create the API instance.
- static std::shared_ptr< API > Create (const std::shared_ptr< Device > &device)

  Create the API instance.

9.1.1 Detailed Description

The API class to communicate with MYNT® EYE device.

9.1.2 Member Typedef Documentation

9.1.2.1 motion_callback_t

```
using mynteye::API::motion_callback_t = std::function<void(const api::MotionData &data)>
```

The api::MotionData callback.

9.1.2.2 stream_callback_t

```
using mynteye::API::stream_callback_t = std::function<void(const api::StreamData &data)>
```

The api::StreamData callback.

9.1.3 Member Function Documentation

9.1.3.1 Create() [1/3]

```
static std::shared_ptr<API> mynteye::API::Create (
            int argc,
            char * argv[] )  [static]
```

Create the API instance.

Parameters

argc	the arg count.
argv	the arg values.

Returns

the API instance.

Note

This will init glog with args and call device::select() to select a device.

9.1.3.2 Create() [2/3]

```
static std::shared_ptr<API> mynteye::API::Create (
            int argc,
            char * argv[],
            const std::shared_ptr< Device > & device )  [static]
```

Create the API instance.

Parameters

argc	the arg count.
argv	the arg values.
device	the selected device.

Returns

the API instance.

Note

This will init glog with args.

9.1.3.3 Create() [3/3]

```
static std::shared_ptr<API> mynteye::API::Create (
            const std::shared_ptr< Device > & device )  [static]
```

Create the API instance.

Parameters

device	the selected device.

Returns

the API instance.

9.1.3.4 EnableStreamData()

```
void mynteye::API::EnableStreamData (
            const Stream & stream )
```

Enable the data of stream.

Note

must enable the stream if it's a synthetic one. This means the stream in not native, the device has the capability to provide this stream, but still support this stream.

9.1.3.5 GetIntrinsics()

```
IntrinsicsPinhole mynteye::API::GetIntrinsics (
            const Stream & stream ) const
```

Deprecated Get the intrinsics (pinhole) of stream.

9.1.3.6 GetStreamDatas()

```
std::vector<api::StreamData> mynteye::API::GetStreamDatas (
            const Stream & stream )
```

Get the datas of stream.

Note

default cache 4 datas at most.

9.2 mynteye::AsyncCallback< Data > Class Template Reference

9.3 mynteye::Context Class Reference

The context about devices.

Public Member Functions

- std::vector< std::shared_ptr< Device > > devices () const

  Get all devices now.

9.3.1 Detailed Description

The context about devices.

9.3.2 Member Function Documentation

9.3.2.1 devices()

```
std::vector<std::shared_ptr<Device> > mynteye::Context::devices ( ) const  [inline]
```

Get all devices now.

Returns

a vector of all devices.

9.4 mynteye::Device Class Reference

The Device class to communicate with MYNT® EYE device.

Public Types

- using stream_callback_t = device::StreamCallback

  The device::StreamData callback.

- using motion_callback_t = device::MotionCallback

  The device::MotionData callback.

Public Member Functions

- Model GetModel () const

  Get the model.
- bool Supports (const Stream &stream) const

  Supports the stream or not.
- bool Supports (const Capabilities &capability) const

  Supports the capability or not.
- bool Supports (const Option &option) const

  Supports the option or not.
- bool Supports (const AddOns &addon) const

  Supports the addon or not.
- const std::vector< StreamRequest > & GetStreamRequests (const Capabilities &capability) const

  Get all stream requests of the capability.
- void ConfigStreamRequest (const Capabilities &capability, const StreamRequest &request)

  Config the stream request to the capability.
- const StreamRequest & GetStreamRequest (const Capabilities &capability) const

  Get the config stream requests of the capability.
- const std::vector< StreamRequest > & GetStreamRequests () const

  Get all stream requests of the key stream capability.
- void ConfigStreamRequest (const StreamRequest &request)

  Config the stream request to the key stream capability.
- const StreamRequest & GetStreamRequest () const

  Get the config stream requests of the key stream capability.
- std::shared_ptr< DeviceInfo > GetInfo () const

  Get the device info.
- std::string GetInfo (const Info &info) const

  Get the device info of a field.
- std::shared_ptr< IntrinsicsBase > GetIntrinsics (const Stream &stream) const

  Get the intrinsics of stream.
- Extrinsics GetExtrinsics (const Stream &from, const Stream &to) const

  Get the extrinsics from one stream to another.
- MotionIntrinsics GetMotionIntrinsics () const

  Get the intrinsics of motion.
- Extrinsics GetMotionExtrinsics (const Stream &from) const

  Get the extrinsics from one stream to motion.
- std::shared_ptr< IntrinsicsBase > GetIntrinsics (const Stream &stream, bool *ok) const

  Get the intrinsics of stream.
- Extrinsics GetExtrinsics (const Stream &from, const Stream &to, bool *ok) const

  Get the extrinsics from one stream to another.
- MotionIntrinsics GetMotionIntrinsics (bool *ok) const

  Get the intrinsics of motion.
- Extrinsics GetMotionExtrinsics (const Stream &from, bool *ok) const

  Get the extrinsics from one stream to motion.
- void SetIntrinsics (const Stream &stream, const std::shared_ptr< IntrinsicsBase > &in)

  Set the intrinsics of stream.
- void SetExtrinsics (const Stream &from, const Stream &to, const Extrinsics &ex)

  Set the extrinsics from one stream to another.
- void SetMotionIntrinsics (const MotionIntrinsics &in)

  Set the intrinsics of motion.
- void SetMotionExtrinsics (const Stream &from, const Extrinsics &ex)

Set the extrinsics from one stream to motion.

- void LogOptionInfos () const

  Log all option infos.
- OptionInfo GetOptionInfo (const Option &option) const

  Get the option info.
- std::int32_t GetOptionValue (const Option &option) const

  Get the option value.
- void SetOptionValue (const Option &option, std::int32_t value)

  Set the option value.
- bool RunOptionAction (const Option &option) const

  Run the option action.
- void SetStreamCallback (const Stream &stream, stream_callback_t callback, bool async=false)

  Set the callback of stream.
- void SetMotionCallback (motion_callback_t callback, bool async=false)

  Set the callback of motion.
- bool HasStreamCallback (const Stream &stream) const

  Has the callback of stream.
- bool HasMotionCallback () const

  Has the callback of motion.
- virtual void Start (const Source &source)

  Start capturing the source.
- virtual void Stop (const Source &source)

  Stop capturing the source.
- void WaitForStreams ()

  Wait the streams are ready.
- device::StreamData GetStreamData (const Stream &stream)

  Get the latest data of stream.
- device::StreamData GetLatestStreamData (const Stream &stream)
- std::vector< device::StreamData > GetStreamDatas (const Stream &stream)

  Get the datas of stream.
- void EnableMotionDatas ()

  Enable cache motion datas.
- void EnableMotionDatas (std::size_t max_size)

  Enable cache motion datas.
- std::vector< device::MotionData > GetMotionDatas ()

  Get the motion datas.

Static Public Member Functions

- static std::shared_ptr< Device > Create (const std::string &name, std::shared_ptr< uvc::device > device)

  Create the Device instance.

9.4.1 Detailed Description

The Device class to communicate with MYNT® EYE device.

9.4.2 Member Typedef Documentation

9.4.2.1 motion_callback_t

```
using mynteye::Device::motion_callback_t = device::MotionCallback
```

The device::MotionData callback.

9.4.2.2 stream_callback_t

```
using mynteye::Device::stream_callback_t = device::StreamCallback
```

The device::StreamData callback.

9.4.3 Member Function Documentation

9.4.3.1 Create()

```
static std::shared_ptr<Device> mynteye::Device::Create (
            const std::string & name,
            std::shared_ptr< uvc::device > device )  [static]
```

Create the Device instance.

Parameters

name	the device name.
device	the device from uvc.

Returns

the Device instance.

9.4.3.2 GetLatestStreamData()

```
device::StreamData mynteye::Device::GetLatestStreamData (
            const Stream & stream )
```

Deprecated Replaced by GetStreamData(const Stream &stream)

9.4.3.3 GetStreamDatas()

```
std::vector<device::StreamData> mynteye::Device::GetStreamDatas (
            const Stream & stream )
```

Get the datas of stream.

Note

default cache 4 datas at most.

9.5 mynteye::DeviceInfo Struct Reference

Device infomation.

9.5.1 Detailed Description

Device infomation.

9.6 mynteye::Extrinsics Struct Reference

Extrinsics, represent how the different datas are connected.

Public Member Functions

- Extrinsics Inverse () const
  Inverse this extrinsics.

Public Attributes

- double rotation [3][3]
  Rotation matrix.
- double translation [3]
  Translation vector.

9.6.1 Detailed Description

Extrinsics, represent how the different datas are connected.

9.6.2 Member Function Documentation

9.6.2.1 Inverse()

```
Extrinsics mynteye::Extrinsics::Inverse ( ) const  [inline]
```

Inverse this extrinsics.

Returns

the inversed extrinsics.

9.7 mynteye::device::Frame Class Reference

Frame with raw data.

Public Member Functions

- Frame (const StreamRequest &request, const void *data)
  Construct the frame with StreamRequest and raw data.
- Frame (std::uint16_t width, std::uint16_t height, Format format, const void *data)
  Construct the frame with stream info and raw data.
- std::uint16_t width () const
  Get the width.
- std::uint16_t height () const
  Get the height.
- Format format () const
  Get the format.
- std::uint8_t * data ()
  Get the data.
- const std::uint8_t * data () const
  Get the const data.
- std::size_t size () const
  Get the size of data.
- Frame clone () const
  Clone a new frame.

9.7.1 Detailed Description

Frame with raw data.

9.7.2 Member Function Documentation

9.7.2.1 clone()

```
Frame mynteye::device::Frame::clone ( ) const  [inline]
```

Clone a new frame.

9.7.2.2 data() [1/2]

```
std::uint8_t* mynteye::device::Frame::data ( )  [inline]
```

Get the data.

9.7.2.3 data() [2/2]

```
const std::uint8_t* mynteye::device::Frame::data ( ) const  [inline]
```

Get the const data.

9.7.2.4 format()

```
Format mynteye::device::Frame::format ( ) const  [inline]
```

Get the format.

9.7.2.5 height()

```
std::uint16_t mynteye::device::Frame::height ( ) const  [inline]
```

Get the height.

9.7.2.6 size()

```
std::size_t mynteye::device::Frame::size ( ) const  [inline]
```

Get the size of data.

9.7.2.7 width()

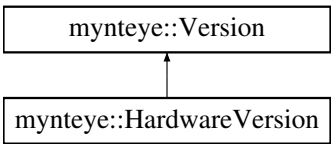
```
std::uint16_t mynteye::device::Frame::width ( ) const  [inline]
```

Get the width.

9.8 mynteye::HardwareVersion Class Reference

Hardware version.

Inheritance diagram for mynteye::HardwareVersion:



9.8.1 Detailed Description

Hardware version.

9.9 mynteye::ImgData Struct Reference

Image data.

Public Attributes

- std::uint16_t frame_id
  Image frame id.
- std::uint64_t timestamp
  Image timestamp in 1us.
- std::uint16_t exposure_time
  Image exposure time, virtual value in [1, 480].

9.9.1 Detailed Description

Image data.

9.10 mynteye::device::ImgParams Struct Reference

9.11 mynteye::ImuData Struct Reference

IMU data.

Public Attributes

- std::uint32_t frame_id

  IMU frame id.
- std::uint8_t flag

  IMU accel or gyro flag.
- std::uint64_t timestamp

  IMU timestamp in 1us.
- double accel [3]

  IMU accelerometer data for 3-axis: X, Y, Z.
- double gyro [3]

  IMU gyroscope data for 3-axis: X, Y, Z.
- double temperature

  IMU temperature.

9.11.1 Detailed Description

IMU data.

9.11.2 Member Data Documentation

9.11.2.1 accel

```
double mynteye::ImuData::accel[3]
```

IMU accelerometer data for 3-axis: X, Y, Z.

9.11.2.2 flag

```
std::uint8_t mynteye::ImuData::flag
```

IMU accel or gyro flag.

0: accel and gyro are both valid 1: accel is valid 2: gyro is valid

9.11.2.3 gyro

```
double mynteye::ImuData::gyro[3]
```

IMU gyroscope data for 3-axis: X, Y, Z.

9.12 mynteye::ImuIntrinsics Struct Reference

IMU intrinsics: scale, drift and variances.

Public Attributes

- double scale [3][3]
  Scale matrix.
- double noise [3]
  Noise density variances.
- double bias [3]
  Random walk variances.

9.12.1 Detailed Description

IMU intrinsics: scale, drift and variances.

9.12.2 Member Data Documentation

9.12.2.1 scale

```
double mynteye::ImuIntrinsics::scale[3][3]
```

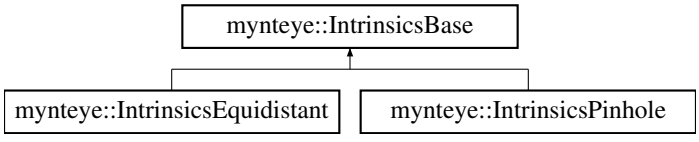
Scale matrix.

```
Scale X     cross axis  cross axis
cross axis  Scale Y     cross axis
cross axis  cross axis  Scale Z
```

9.13 mynteye::device::ImuParams Struct Reference

9.14 mynteye::IntrinsicsBase Struct Reference

Inheritance diagram for mynteye::IntrinsicsBase:



Public Member Functions

- CalibrationModel calib_model () const
  The calibration model.

Public Attributes

- std::uint16_t width

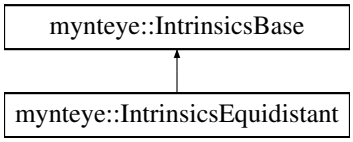
  The width of the image in pixels.
- std::uint16_t height

  The height of the image in pixels.

9.15 mynteye::IntrinsicsEquidistant Struct Reference

Stream intrinsics (Equidistant: KANNALA_BRANDT)

Inheritance diagram for mynteye::IntrinsicsEquidistant:



Public Attributes

- double coeffs [8]

  The distortion coefficients: k2,k3,k4,k5,mu,mv,u0,v0.

Additional Inherited Members

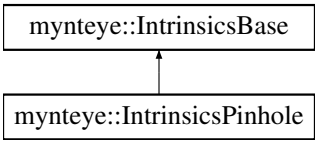
9.15.1 Detailed Description

Stream intrinsics (Equidistant: KANNALA_BRANDT)

9.16 mynteye::IntrinsicsPinhole Struct Reference

Stream intrinsics (Pinhole)

Inheritance diagram for mynteye::IntrinsicsPinhole:



Public Attributes

- double fx
  The focal length of the image plane, as a multiple of pixel width.
- double fy
  The focal length of the image plane, as a multiple of pixel height.
- double cx
  The horizontal coordinate of the principal point of the image.
- double cy
  The vertical coordinate of the principal point of the image.
- std::uint8_t model
- double coeffs [5]
  The distortion coefficients: k1,k2,p1,p2,k3.

Additional Inherited Members

9.16.1 Detailed Description

Stream intrinsics (Pinhole)

9.16.2 Member Data Documentation

9.16.2.1 model

```
std::uint8_t mynteye::IntrinsicsPinhole::model
```

Deprecated Replaced by calib_model_.

The distortion model of the image

9.17 mynteye::device::MotionData Struct Reference

Device motion data.

Public Attributes

- std::shared_ptr< ImuData > imu
  ImuData.

9.17.1 Detailed Description

Device motion data.

9.17.2 Member Data Documentation

9.17.2.1 imu

```
std::shared_ptr<ImuData> mynteye::device::MotionData::imu
```

ImuData.

9.18 mynteye::api::MotionData Struct Reference

API motion data.

Public Attributes

- std::shared_ptr< ImuData > imu
  ImuData.

9.18.1 Detailed Description

API motion data.

9.18.2 Member Data Documentation

9.18.2.1 imu

```
std::shared_ptr<ImuData> mynteye::api::MotionData::imu
```

ImuData.

9.19 mynteye::MotionIntrinsics Struct Reference

Motion intrinsics, including accelerometer and gyroscope.

Public Attributes

- ImuIntrinsics accel
  Accelerometer intrinsics.
- ImuIntrinsics gyro
  Gyroscope intrinsics.

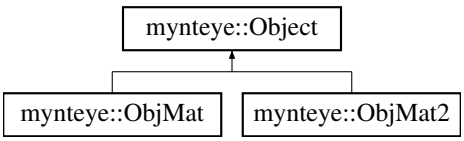
9.19.1 Detailed Description

Motion intrinsics, including accelerometer and gyroscope.

9.20 mynteye::Object Struct Reference

Input & output object.

Inheritance diagram for mynteye::Object:



Static Public Member Functions

- template<typename T >
  static T * Cast (Object *obj)
  Cast the obj to T pointer.
- template<typename T >
  static const T * Cast (const Object *obj)
  Cast the obj to const T pointer.

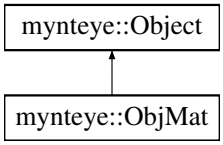
9.20.1 Detailed Description

Input & output object.

9.21 mynteye::ObjMat Struct Reference

Input & output object of one cv::Mat.

Inheritance diagram for mynteye::ObjMat:



Public Attributes

- cv::Mat value
  The value.
- std::uint16_t id
  The id.
- std::shared_ptr< ImgData > data
  The data.

Additional Inherited Members

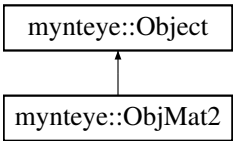
9.21.1 Detailed Description

Input & output object of one cv::Mat.

9.22 mynteye::ObjMat2 Struct Reference

Input & output object of two cv::Mat.

Inheritance diagram for mynteye::ObjMat2:



Public Attributes

- cv::Mat first
  The first value.
- std::uint16_t first_id
  The first id.
- std::shared_ptr< ImgData > first_data
  The first data.
- cv::Mat second
  The second value.
- std::uint16_t second_id
  The second id.
- std::shared_ptr< ImgData > second_data
  The second data.

Additional Inherited Members

9.22.1 Detailed Description

Input & output object of two cv::Mat.

9.23 mynteye::OptionInfo Struct Reference

Option info.

Public Attributes

- std::int32_t min
  Minimum value.
- std::int32_t max
  Maximum value.
- std::int32_t def
  Default value.

9.23.1 Detailed Description

Option info.

9.24 mynteye::Plugin Class Reference

The plugin which could implement processing by yourself.

Public Member Functions

- virtual void OnCreate (API *api)
  Called when plugin created.
- virtual bool OnRectifyProcess (Object *const in, Object *const out)
  Called when process rectify.
- virtual bool OnDisparityProcess (Object *const in, Object *const out)
  Called when process disparity.
- virtual bool OnDisparityNormalizedProcess (Object *const in, Object *const out)
  Called when process normalized disparity.
- virtual bool OnPointsProcess (Object *const in, Object *const out)
  Called when process points.
- virtual bool OnDepthProcess (Object *const in, Object *const out)
  Called when process depth.

9.24.1 Detailed Description

The plugin which could implement processing by yourself.

9.24.2 Member Function Documentation

9.24.2.1 OnCreate()

```
virtual void mynteye::Plugin::OnCreate (
            API * api )  [inline], [virtual]
```

Called when plugin created.

Parameters

api	the API instacne.

9.24.2.2 OnDepthProcess()

```
virtual bool mynteye::Plugin::OnDepthProcess (
            Object *const in,
            Object *const out )  [inline], [virtual]
```

Called when process depth.

Parameters

in	input object.
out	output object.

Returns

`true` if you process depth.

9.24.2.3 OnDisparityNormalizedProcess()

```
virtual bool mynteye::Plugin::OnDisparityNormalizedProcess (
            Object *const in,
            Object *const out )  [inline], [virtual]
```

Called when process normalized disparity.

Parameters

in	input object.
out	output object.

Returns

`true` if you process normalized disparity.

9.24.2.4 OnDisparityProcess()

```
virtual bool mynteye::Plugin::OnDisparityProcess (
            Object *const in,
            Object *const out )  [inline], [virtual]
```

Called when process disparity.

Parameters

in	input object.
out	output object.

Returns

`true` if you process disparity.

9.24.2.5 OnPointsProcess()

```
virtual bool mynteye::Plugin::OnPointsProcess (
            Object *const in,
            Object *const out )  [inline], [virtual]
```

Called when process points.

Parameters

in	input object.
out	output object.

Returns

`true` if you process points.

9.24.2.6 OnRectifyProcess()

```
virtual bool mynteye::Plugin::OnRectifyProcess (
            Object *const in,
            Object *const out )  [inline], [virtual]
```

Called when process rectify.

Parameters

in	input object.
out	output object.

Returns

`true` if you process rectify.

9.25 mynteye::Resolution Struct Reference

Resolution.

Public Attributes

- std::uint16_t width
  Width.
- std::uint16_t height
  Height.

9.25.1 Detailed Description

Resolution.

9.26 mynteye::api::StreamData Struct Reference

API stream data.

Public Attributes

- std::shared_ptr< ImgData > img
  ImgData.
- cv::Mat frame
  Frame.
- std::shared_ptr< device::Frame > frame_raw
  Raw frame.
- std::uint16_t frame_id
  Frame ID.

9.26.1 Detailed Description

API stream data.

9.26.2 Member Data Documentation

9.26.2.1 frame

```
cv::Mat mynteye::api::StreamData::frame
```

Frame.

9.26.2.2 frame_id

```
std::uint16_t mynteye::api::StreamData::frame_id
```

Frame ID.

9.26.2.3 frame_raw

```
std::shared_ptr<device::Frame> mynteye::api::StreamData::frame_raw
```

Raw frame.

9.26.2.4 img

```
std::shared_ptr<ImgData> mynteye::api::StreamData::img
```

ImgData.

9.27 mynteye::device::StreamData Struct Reference

Device stream data.

Public Attributes

- std::shared_ptr< ImgData > img
  ImgData.
- std::shared_ptr< Frame > frame
  Frame.
- std::uint16_t frame_id
  Frame ID.

9.27.1 Detailed Description

Device stream data.

9.27.2 Member Data Documentation

9.27.2.1 frame

```
std::shared_ptr<Frame> mynteye::device::StreamData::frame
```

Frame.

9.27.2.2 frame_id

```
std::uint16_t mynteye::device::StreamData::frame_id
```

Frame ID.

9.27.2.3 img

```
std::shared_ptr<ImgData> mynteye::device::StreamData::img
```

ImgData.

9.28 mynteye::StreamRequest Struct Reference

Stream request.

Public Attributes

- std::uint16_t width
  Stream width in pixels.
- std::uint16_t height
  Stream height in pixels.
- Format format
  Stream pixel format.
- std::uint16_t fps
  Stream frames per second.

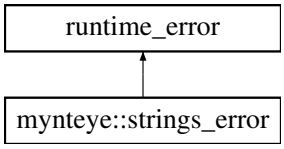
9.28.1 Detailed Description

Stream request.

9.29 mynteye::strings_error Class Reference

The strings error.

Inheritance diagram for mynteye::strings_error:



9.29.1 Detailed Description

The strings error.

9.30 mynteye::Type Class Reference

Type.

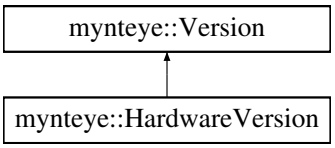
9.30.1 Detailed Description

Type.

9.31 mynteye::Version Class Reference

Version.

Inheritance diagram for mynteye::Version:



9.31.1 Detailed Description

Version.

Index