/\*\*

@copyright COPYRIGHT 2017 - PROPERTY OF TOBII AB

@copyright 2017 TOBII AB - KARLSROVAGEN 2D, DANDERYD 182 53, SWEDEN - All Rights Reserved.

@copyright NOTICE: All information contained herein is, and remains, the property of Tobii AB and its suppliers, if any. The intellectual and technical concepts contained herein are proprietary to Tobii AB and its suppliers and may be covered by U.S.and Foreign Patents, patent applications, and are protected by trade secret or copyright law. Dissemination of this information or reproduction of this material is strictly forbidden unless prior written permission is obtained from Tobii AB.

\*/

/\*\*

\* @file tobii\_research\_streams.h

\* @brief <b>Functionality for eye tracker streams.</b>

\*

\*/

#ifndef TOBII\_RESEARCH\_STREAMS\_H\_

#define TOBII\_RESEARCH\_STREAMS\_H\_

#include "tobii\_research.h"

#include "tobii\_research\_eyetracker.h"

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*\*

Specifies the validity.

\*/

typedef enum {

/\*\*

Indicates invalid.

\*/

TOBII\_RESEARCH\_VALIDITY\_INVALID,

/\*\*

Indicates valid.

\*/

TOBII\_RESEARCH\_VALIDITY\_VALID

} TobiiResearchValidity;

/\*\*

Provides properties for the gaze origin.

\*/

typedef struct {

/\*\*

The gaze origin position in 3D in the user coordinate system.

\*/

TobiiResearchPoint3D position\_in\_user\_coordinates;

/\*\*

The normalized gaze origin position in 3D in the track box coordinate system.

\*/

TobiiResearchNormalizedPoint3D position\_in\_track\_box\_coordinates;

/\*\*

The validity of the gaze origin data.

\*/

TobiiResearchValidity validity;

} TobiiResearchGazeOrigin;

/\*\*

Provides properties for the pupil data.

\*/

typedef struct {

/\*\*

The diameter of the pupil in millimeters.

\*/

float diameter;

/\*\*

The validity of the pupil data.

\*/

TobiiResearchValidity validity;

} TobiiResearchPupilData;

/\*\*

Provides properties for the gaze point.

\*/

typedef struct {

/\*\*

The gaze point position in 2D on the active display area.

\*/

TobiiResearchNormalizedPoint2D position\_on\_display\_area;

/\*\*

The gaze point position in 3D in the user coordinate system.

\*/

TobiiResearchPoint3D position\_in\_user\_coordinates;

/\*\*

The validity of the gaze point data.

\*/

TobiiResearchValidity validity;

} TobiiResearchGazePoint;

/\*\*

Provides properties for the eye data.

\*/

typedef struct {

/\*\*

The gaze point data.

\*/

TobiiResearchGazePoint gaze\_point;

/\*\*

The pupil data.

\*/

TobiiResearchPupilData pupil\_data;

/\*\*

The gaze origin data.

\*/

TobiiResearchGazeOrigin gaze\_origin;

} TobiiResearchEyeData;

/\*\*

Provides data for the @ref tobii\_research\_gaze\_data\_callback callback.

\*/

typedef struct {

/\*\*

The gaze data for the left eye.

\*/

TobiiResearchEyeData left\_eye;

/\*\*

The gaze data for the right eye.

\*/

TobiiResearchEyeData right\_eye;

/\*\*

The time stamp according to the eye tracker's internal clock.

\*/

int64\_t device\_time\_stamp;

/\*\*

The time stamp according to the computer's internal clock.

\*/

int64\_t system\_time\_stamp;

} TobiiResearchGazeData;

/\*\*

Provides properties for the HMD pupil position.

\*/

typedef struct {

/\*\*

The (normalized) 2D coordinates that describes the pupil's position in the HMD's tracking area.

\*/

TobiiResearchNormalizedPoint2D position\_in\_tracking\_area;

/\*\*

The validity of the pupil position data.

\*/

TobiiResearchValidity validity;

} TobiiResearchHMDPupilPosition;

/\*\*

Provides properties for the HMD gaze origin.

\*/

typedef struct {

/\*\*

The 3D coordinates that describes the gaze origin in (in mm).

\*/

TobiiResearchPoint3D position\_in\_hmd\_coordinates;

/\*\*

The validity of the gaze origin data.

\*/

TobiiResearchValidity validity;

} TobiiResearchHMDGazeOrigin;

/\*\*

Provides properties for the HMD gaze direction.

\*/

typedef struct {

/\*\*

The 3D unit vector that describes the gaze direction.

\*/

TobiiResearchNormalizedPoint3D unit\_vector;

/\*\*

The validity of the gaze direction data.

\*/

TobiiResearchValidity validity;

} TobiiResearchHMDGazeDirection;

/\*\*

Provides properties for the eye data when gotten from an HMD based device.

\*/

typedef struct {

/\*\*

The gaze direction data.

\*/

TobiiResearchHMDGazeDirection gaze\_direction;

/\*\*

The pupil data.

\*/

TobiiResearchPupilData pupil\_data;

/\*\*

The gaze origin data.

\*/

TobiiResearchHMDGazeOrigin gaze\_origin;

/\*\*

The pupil position in HMD track box.

\*/

TobiiResearchHMDPupilPosition pupil\_position;

} TobiiResearchHMDEyeData;

/\*\*

Provides data for the @ref tobii\_research\_hmd\_gaze\_data\_callback callback.

\*/

typedef struct {

/\*\*

The gaze data for the left eye.

\*/

TobiiResearchHMDEyeData left\_eye;

/\*\*

The gaze data for the right eye.

\*/

TobiiResearchHMDEyeData right\_eye;

/\*\*

The time stamp according to the eye tracker's internal clock.

\*/

int64\_t device\_time\_stamp;

/\*\*

The time stamp according to the computer's internal clock.

\*/

int64\_t system\_time\_stamp;

} TobiiResearchHMDGazeData;

/\*\*

Provides data for the @ref tobii\_research\_time\_synchronization\_data\_callback callback.

\*/

typedef struct {

/\*\*

The time stamp when the computer sent the request to the eye tracker.

\*/

int64\_t system\_request\_time\_stamp;

/\*\*

The time stamp when the eye tracker received the request, according to the eye tracker's clock.

\*/

int64\_t device\_time\_stamp;

/\*\*

The time stamp when the computer received the response from the eye tracker.

\*/

int64\_t system\_response\_time\_stamp;

} TobiiResearchTimeSynchronizationData;

/\*\*

Defines external signal change type.

\*/

typedef enum {

/\*\*

Indicates that the value sent to the eye tracker has changed.

\*/

TOBII\_RESEARCH\_EXTERNAL\_SIGNAL\_VALUE\_CHANGED,

/\*\*

Indicates that the value is the initial value, and is received when starting a subscription.

\*/

TOBII\_RESEARCH\_EXTERNAL\_SIGNAL\_INITIAL\_VALUE,

/\*\*

Indicates that there has been a connection lost and now it is restored and the value is the current value.

\*/

TOBII\_RESEARCH\_EXTERNAL\_SIGNAL\_CONNECTION\_RESTORED

} TobiiResearchExternalSignalChangeType;

/\*\*

Provides data for the @ref tobii\_research\_external\_signal\_data\_callback callback.

\*/

typedef struct {

/\*\*

The time stamp according to the eye tracker's internal clock.

\*/

int64\_t device\_time\_stamp;

/\*\*

The time stamp according to the computer's internal clock.

\*/

int64\_t system\_time\_stamp;

/\*\*

The value of the external signal port on the eye tracker.

\*/

uint32\_t value;

TobiiResearchExternalSignalChangeType change\_type;

} TobiiResearchExternalSignalData;

/\*\*

Defines error types

\*/

typedef enum {

/\*\*

Indicates that the connection to the eye tracker was lost.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_CONNECTION\_LOST,

/\*\*

Indicates that the license is insufficient for subscribing to the stream.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_INSUFFICIENT\_LICENSE,

/\*\*

Indicates that the stream isn't supported by the eye tracker.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_NOT\_SUPPORTED,

/\*\*

Indicates that an internal error occurred.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_INTERNAL\_ERROR,

/\*\*

Indicates that the user threw an exception in the callback.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_USER\_ERROR

} TobiiResearchStreamError;

/\*\*

Defines error sources

\*/

typedef enum {

/\*\*

User callback failed.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_SOURCE\_USER,

/\*\*

Error when pumping event.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_SOURCE\_STREAM\_PUMP,

/\*\*

Error when subscribing to event for gaze data.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_SOURCE\_SUBSCRIPTION\_GAZE\_DATA,

/\*\*

Error when subscribing to event for external signal.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_SOURCE\_SUBSCRIPTION\_EXTERNAL\_SIGNAL,

/\*\*

Error when subscribing to event for time synchronization data.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_SOURCE\_SUBSCRIPTION\_TIME\_SYNCHRONIZATION\_DATA,

/\*\*

Error when subscribing to event for eye images.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_SOURCE\_SUBSCRIPTION\_EYE\_IMAGE,

/\*\*

Error when subscribing to notification event.

\*/

TOBII\_RESEARCH\_STREAM\_ERROR\_SOURCE\_SUBSCRIPTION\_NOTIFICATION

} TobiiResearchStreamErrorSource;

/\*\*

Defines notification types

\*/

typedef enum {

/\*\*

Indicates that the connection to the eye tracker is lost

\*/

TOBII\_RESEARCH\_NOTIFICATION\_CONNECTION\_LOST,

/\*\*

Indicates that the connection to the eye tracker is restored

\*/

TOBII\_RESEARCH\_NOTIFICATION\_CONNECTION\_RESTORED,

/\*\*

Indicates that the calibration mode is entered.

\*/

TOBII\_RESEARCH\_NOTIFICATION\_CALIBRATION\_MODE\_ENTERED,

/\*\*

Indicates that the calibration mode is left.

\*/

TOBII\_RESEARCH\_NOTIFICATION\_CALIBRATION\_MODE\_LEFT,

/\*\*

Indicates that the track box is changed.

\*/

TOBII\_RESEARCH\_NOTIFICATION\_TRACK\_BOX\_CHANGED,

/\*\*

Indicates that the display area is changed.

\*/

TOBII\_RESEARCH\_NOTIFICATION\_DISPLAY\_AREA\_CHANGED,

/\*\*

Indicates that the gaze output frequency is changed

\*/

TOBII\_RESEARCH\_NOTIFICATION\_GAZE\_OUTPUT\_FREQUENCY\_CHANGED,

/\*\*

\*/

TOBII\_RESEARCH\_NOTIFICATION\_UNKNOWN

} TobiiResearchNotificationType;

/\*\*

Provides data for the @ref tobii\_research\_notification\_callback callback.

\*/

typedef struct {

/\*\*

The time stamp according to the eye tracker's internal clock.

\*/

int64\_t system\_time\_stamp;

/\*\*

The notification type.

\*/

TobiiResearchNotificationType notification\_type;

union {

/\*\*

The new output frequency if notification of type @ref TOBII\_RESEARCH\_NOTIFICATION\_GAZE\_OUTPUT\_FREQUENCY\_CHANGED

\*/

float output\_frequency;

/\*\*

The new display area if notification of type @ref TOBII\_RESEARCH\_NOTIFICATION\_DISPLAY\_AREA\_CHANGED

\*/

TobiiResearchDisplayArea display\_area;

} value;

} TobiiResearchNotification;

/\*\*

@brief Gaze data callback.

Implement this and send as a parameter to @ref tobii\_research\_subscribe\_to\_gaze\_data.

@param gaze\_data: Gaze data received from the eye tracker.

@param user\_data: Caller specific data sent in with @ref tobii\_research\_subscribe\_to\_gaze\_data.

\*/

typedef void(\*tobii\_research\_gaze\_data\_callback)(TobiiResearchGazeData\* gaze\_data, void\* user\_data);

/\*\*

@brief Subscribes to gaze data for the eye tracker.

You will get a callback when time synchronized gaze is received.Time synchronized gaze is not supported on all eye trackers,

other eye trackers need additional license to activate this support.

\snippet gaze\_data.c Example

@param eyetracker: Eye tracker object.

@param callback: Callback that will receive the gaze data.

@param user\_data: Caller specific data that will be sent to the callback.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_subscribe\_to\_gaze\_data(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_gaze\_data\_callback callback, void\* user\_data);

/\*\*

@brief Unsubscribes from gaze data for the eye tracker.

@param eyetracker: Eye tracker object.

@param callback: Callback sent to @ref tobii\_research\_subscribe\_to\_gaze\_data.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_unsubscribe\_from\_gaze\_data(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_gaze\_data\_callback callback);

/\*\*

@brief HMD gaze data callback.

Implement this and send as a parameter to @ref tobii\_research\_subscribe\_to\_hmd\_gaze\_data.

@param hmd\_gaze\_data: HMD gaze data received from the eye tracker.

@param user\_data: Caller specific data sent in with @ref tobii\_research\_subscribe\_to\_hmd\_gaze\_data.

\*/

typedef void(\*tobii\_research\_hmd\_gaze\_data\_callback)(TobiiResearchHMDGazeData\* hmd\_gaze\_data, void\* user\_data);

/\*\*

@brief Subscribes to gaze data for the eye tracker.

@param eyetracker: Eye tracker object.

@param callback: Callback that will receive the gaze data.

@param user\_data: Caller specific data that will be sent to the callback.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_subscribe\_to\_hmd\_gaze\_data(

TobiiResearchEyeTracker\* eyetracker, tobii\_research\_hmd\_gaze\_data\_callback callback, void\* user\_data);

/\*\*

@brief Unsubscribes from HMD gaze data for the eye tracker.

@param eyetracker: Eye tracker object.

@param callback: Callback sent to @ref tobii\_research\_subscribe\_to\_hmd\_gaze\_data.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_unsubscribe\_from\_hmd\_gaze\_data(

TobiiResearchEyeTracker\* eyetracker, tobii\_research\_hmd\_gaze\_data\_callback callback);

/\*\*

@brief External signal callback.

Implement this and send as a parameter to #tobii\_research\_subscribe\_to\_external\_signal\_data.

@param external\_signal\_data: External signal data received from the eye tracker.

@param user\_data: Caller specific data sent in with #tobii\_research\_subscribe\_to\_external\_signal\_data.

\*/

typedef void(\*tobii\_research\_external\_signal\_data\_callback)(TobiiResearchExternalSignalData\* external\_signal\_data,

void\* user\_data);

/\*\*

@brief Subscribes to external signal data for the eye tracker.

You will get a callback when the value of the external signal port (TTL input) on the eye tracker device changes. Not

all eye trackers have an output trigger port. The output feature could be used to synchronize the eye tracker data

with data from other devices. The output data contains a time reference that matches the time reference on the time

synchronized gaze data.

\snippet external\_signal.c Example

@param eyetracker: Eye tracker object.

@param callback: Callback that will receive the external signal data.

@param user\_data: Caller specific data that will be sent to the callback.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_subscribe\_to\_external\_signal\_data(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_external\_signal\_data\_callback callback, void\* user\_data);

/\*\*

@brief Unsubscribes from external signal data for the eye tracker.

@param eyetracker: Eye tracker object.

@param callback: Callback sent to @ref tobii\_research\_subscribe\_to\_external\_signal\_data

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_unsubscribe\_from\_external\_signal\_data(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_external\_signal\_data\_callback callback);

/\*\*

@brief Time synchronization callback.

Implement this and send as a parameter to @ref tobii\_research\_subscribe\_to\_time\_synchronization\_data.

@param gaze\_data: Time synchronization data received from the eye tracker.

@param user\_data: Caller specific data sent in with @ref tobii\_research\_subscribe\_to\_time\_synchronization\_data.

\*/

typedef void(\*tobii\_research\_time\_synchronization\_data\_callback)(

TobiiResearchTimeSynchronizationData\* time\_synchronization\_data,

void\* user\_data);

/\*\*

@brief Subscribes to time synchronization data for the eye tracker.

You will get a callback when the computer and the eye trackers clocks gets synchronized. To handle normal drifts

between clocks the clocks are checked on regular basis, and this results in that the time stamps are adjusted for the

drifts in the data streams. This drift handling is done in real time. The data received from this event could be used

for an even more accurate drift adjustment in the post processing.

\snippet time\_synchronization\_data.c Example

@param eyetracker: Eye tracker object.

@param callback: Callback that will receive the time synchronization data.

@param user\_data: Caller specific data that will be sent to the callback.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_subscribe\_to\_time\_synchronization\_data(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_time\_synchronization\_data\_callback callback,

void\* user\_data);

/\*\*

@brief Unsubscribes from time synchronization data for the eye tracker.

@param eyetracker: Eye tracker object.

@param callback: Callback sent to @ref tobii\_research\_subscribe\_to\_time\_synchronization\_data.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_unsubscribe\_from\_time\_synchronization\_data(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_time\_synchronization\_data\_callback callback);

/\*\*

Provides properties for the stream error data.

\*/

typedef struct {

/\*\*

The time stamp according to the eye tracker's internal clock.

\*/

int64\_t system\_time\_stamp;

/\*\*

Type of error.

\*/

TobiiResearchStreamError error;

/\*\*

Source of the error.

\*/

TobiiResearchStreamErrorSource source;

/\*\*

The error message.

\*/

const char\* message;

} TobiiResearchStreamErrorData;

/\*\*

@brief Stream error callback.

Implement this and send as a parameter to @ref tobii\_research\_subscribe\_to\_stream\_errors.

@param stream\_error: @ref TobiiResearchStreamErrorData object.

@param user\_data: Caller specific data sent in with @ref tobii\_research\_subscribe\_to\_stream\_errors.

\*/

typedef void(\*tobii\_research\_stream\_error\_callback)(

TobiiResearchStreamErrorData\* stream\_error,

void\* user\_data);

/\*\*

@brief Subscribes to stream errors for the eye tracker.

You will get a callback when an error occurs on other streams. You can get errors when subscribing, when something

happened to the connection in the stream pump or when an error was raised in a callback.

@param eyetracker: Eye tracker object.

@param callback: Callback that will receive the stream errors.

@param user\_data: Caller specific data that will be sent to the callback.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_subscribe\_to\_stream\_errors(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_stream\_error\_callback callback,

void\* user\_data);

/\*\*

@brief Unsubscribes from stream errors for the eye tracker.

@param eyetracker: Eye tracker object.

@param callback: Callback sent to @ref tobii\_research\_subscribe\_to\_stream\_errors.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_unsubscribe\_from\_stream\_errors(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_stream\_error\_callback callback);

/\*\*

@brief Notification callback.

Implement this and send as a parameter to @ref tobii\_research\_subscribe\_to\_notifications.

@param notification: Notification received from the eye tracker.

@param user\_data: Caller specific data sent in with @ref tobii\_research\_subscribe\_to\_notifications.

\*/

typedef void(\*tobii\_research\_notification\_callback)(

TobiiResearchNotification\* notification,

void\* user\_data);

/\*\*

@brief Subscribes to notifications for the eye tracker.

You will get a callback when notification is received.

\snippet notifications.c Example

@param eyetracker: Eye tracker object.

@param callback: Callback that will receive the notifications.

@param user\_data: Caller specific data that will be sent to the callback.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_subscribe\_to\_notifications(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_notification\_callback callback,

void\* user\_data);

/\*\*

@brief Unsubscribes from notifications for the eye tracker.

@param eyetracker: Eye tracker object.

@param callback: Callback sent to @ref tobii\_research\_subscribe\_to\_notifications.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_unsubscribe\_from\_notifications(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_notification\_callback callback);

/\*\*

Defines eye image type.

\*/

typedef enum {

/\*\*

Indicates that the eye tracker could not identify the eyes, and the image is the full image.

\*/

TOBII\_RESEARCH\_EYE\_IMAGE\_TYPE\_FULL,

/\*\*

Indicates that the image is cropped and shows the eyes.

\*/

TOBII\_RESEARCH\_EYE\_IMAGE\_TYPE\_CROPPED,

/\*\*

The eye image has an unexpected type.

\*/

TOBII\_RESEARCH\_EYE\_IMAGE\_TYPE\_UNKNOWN

} TobiiResearchEyeImageType;

/\*\*

Provides data for the @ref tobii\_research\_eye\_image\_callback callback.

\*/

typedef struct {

/\*\*

The time stamp according to the eye tracker's internal clock.

\*/

int64\_t device\_time\_stamp;

/\*\*

The time stamp according to the computer's internal clock.

\*/

int64\_t system\_time\_stamp;

/\*\*

The bits per pixel for the eye image.

\*/

int bits\_per\_pixel;

/\*\*

The padding bits per pixel for the eye image.

\*/

int padding\_per\_pixel;

/\*\*

The width in pixel for the eye image.

\*/

int width;

/\*\*

The height in pixels for the eye image.

\*/

int height;

/\*\*

The type of eye image.

\*/

TobiiResearchEyeImageType type;

/\*\*

The source/which camera that generated the image.

\*/

int camera\_id;

/\*\*

Size in bytes of the data blob.

\*/

size\_t data\_size;

/\*\*

The data blob sent by the eye tracker.

\*/

void\* data;

} TobiiResearchEyeImage;

/\*\*

Provides data for the @ref tobii\_research\_eye\_image\_as\_gif\_callback callback.

\*/

typedef struct {

/\*\*

The time stamp according to the eye tracker's internal clock.

\*/

int64\_t device\_time\_stamp;

/\*\*

The time stamp according to the computer's internal clock.

\*/

int64\_t system\_time\_stamp;

/\*\*

The type of eye image.

\*/

TobiiResearchEyeImageType type;

/\*\*

The source/which camera that generated the image.

\*/

int camera\_id;

/\*\*

Size in bytes of the image data.

\*/

size\_t image\_size;

/\*\*

The GIF image data.

\*/

void\* image\_data;

} TobiiResearchEyeImageGif;

/\*\*

@brief Eye image callback.

Implement this and send as a parameter to @ref tobii\_research\_subscribe\_to\_eye\_image.

@param frame: Eye image frame received from the eye tracker.

@param user\_data: Caller specific data sent in with @ref tobii\_research\_subscribe\_to\_eye\_image.

\*/

typedef void(\*tobii\_research\_eye\_image\_callback)(TobiiResearchEyeImage\* frame, void\* user\_data);

/\*\*

@brief Eye image gif callback.

Implement this and send as a parameter to @ref tobii\_research\_subscribe\_to\_eye\_image\_as\_gif.

@param frame: Eye image gif frame received from the eye tracker.

@param user\_data: Caller specific data sent in with @ref tobii\_research\_subscribe\_to\_eye\_image\_as\_gif.

\*/

typedef void(\*tobii\_research\_eye\_image\_as\_gif\_callback)(TobiiResearchEyeImageGif\* frame, void\* user\_data);

/\*\*

@brief Subscribes to eye image for the eye tracker.

You will get a callback when a new eye image is received, and the occurrence depends on the eye tracker model. Not all

eye tracker models support this feature. If no one is listening to gaze data, the eye tracker will only deliver full

images, otherwise either cropped or full images will be delivered depending on whether or not the eye tracker has

detected eyes.

@param eyetracker: Eye tracker object.

@param callback: Callback that will receive the frames.

@param user\_data: Caller specific data that will be sent to the callback.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_subscribe\_to\_eye\_image(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_eye\_image\_callback callback,

void\* user\_data);

/\*\*

@brief Subscribes to eye image for the eye tracker with the image delivered in gif format.

You will get a callback when a new eye image is received, and the occurrence depends on the eye tracker model. Not all

eye tracker models support this feature. If no one is listening to gaze data, the eye tracker will only deliver full

images, otherwise either cropped or full images will be delivered depending on whether or not the eye tracker has

detected eyes.

@param eyetracker: Eye tracker object.

@param callback: Callback that will receive the frames.

@param user\_data: Caller specific data that will be sent to the callback.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_subscribe\_to\_eye\_image\_as\_gif(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_eye\_image\_as\_gif\_callback callback,

void\* user\_data);

/\*\*

@brief Unsubscribes from eye image for the eye tracker.

@param eyetracker: Eye tracker object.

@param callback: Callback sent to @ref tobii\_research\_subscribe\_to\_eye\_image.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_unsubscribe\_from\_eye\_image(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_eye\_image\_callback callback);

/\*\*

@brief Unsubscribes from eye image for the eye tracker.

@param eyetracker: Eye tracker object.

@param callback: Callback sent to @ref tobii\_research\_subscribe\_to\_eye\_image\_as\_gif.

@returns A @ref TobiiResearchStatus code.

\*/

TOBII\_RESEARCH\_API TobiiResearchStatus TOBII\_RESEARCH\_CALL tobii\_research\_unsubscribe\_from\_eye\_image\_as\_gif(

TobiiResearchEyeTracker\* eyetracker,

tobii\_research\_eye\_image\_as\_gif\_callback callback);

#ifdef \_\_cplusplus

}

#endif

#endif /\* TOBII\_RESEARCH\_STREAMS\_H\_ \*/