Maestro C API

v0.1 Alpha

Generated by Doxygen 1.8.13

Contents

Chapter 1

Main Page

Introduction

The Maestro C API allows developers to build tools and applications that utilize the Maestro glove's motion capture and haptic functionality.

Terminology

The documentation and SDK refer to different joints in the hand and finger using the terminology depicted above. The first joint on the thumb (where the thumb connects to the palm) has both a proximal rotation that controls inward rotation, much like the finger proximal joints, as well as an abduction rotation that controls the side-to-side rotation of the thumb. These two rotations paired together allow for the circular rotation of the thumb.

Usage

Initial Setup

There are configuration files that must be in place for the API to function properly. These are copied to the correct locations by the Maestro installer (be sure to run the installer associated with the API version you're using).

Configuration

The configuration files for the Maestro can be found in `%LOCALAPPDATA%\Contact Control Interfaces\Maestro\[VERSION]\Configuration` where `[VERSION]` is the version of the SDK that you're working with. For the 0.1 Alpha version, the configuration is located at `%LOCALAPPDATA%\Contact Control Interfaces\Maestro\v0.1a\Configuration`.

Inside of the configuration directory, you'll find two files: `left_maestro_rotation_ranges.txt` and `right_maestro_rotation_ranges.txt`. In most cases the contents of these two files should be the same, but you're able to configure the rotation ranges per-glove if desired. The contents of the rotation range files are used for calculating the rotation of individual joints.

Each line in the file represents a single joint with a comma delimited string of three values: the name of the joint, the minimum rotation, and the maximum rotation. The rotations themselves are specified as positive or negative floating-point numbers in the inclusive range `(0, 1)`, where `0` is no rotation, and `1` is a complete rotation (360 degrees). An example of the contents of one of these rotation range files can be seen below:

```
Wrist,-0.15,0.12
WristAbduction,-0.1,0.1
IndexProximal,-0.02,0.3
IndexDistal,0.0,0.261
MiddleProximal,-0.02,0.3
MiddleDistal,0.0,0.2
RingProximal,-0.045,0.3
RingDistal,0.0,0.2
LittleProximal,-0.045,0.3
LittleDistal,0,0.2
ThumbMetacarpal,-0.06,0.1
ThumbAbduction,-1.0,1.0
ThumbDistal,-0.05,0.2
```

Development

To use the API, you must include the shared library and include the provided header files.

1. The Maestro detection thread must be running for plugged in gloves to be detected. This is as simple as calling `start_maestro_detection_service()`.
2. Now that the service is running, you can use `is_glove_connected(intptr_t maestroPtr)` to determine if a glove is connected. Pass in the appropriate glove pointer obtained by `get_left_glove_pointer()` or `get_right_glove_pointer()`.
3. Once a glove is connected, the glove must be calibrated before the motion capture data can be used. See `Calibration`

Once the Maestro is calibrated, you may retrieve motion capture data for your use in your application, as well as control the vibration and force-feedback haptic systems. For details, view the appropriate modules in the navigation on the left of the page.

Here's a single C program that continously outputs the wrist rotation:

```c
#include <stdio.h>
#include "maestro.h"

bool calibrate_wrist(intptr_t rightGlove)
{
    bool result = false;

    printf("Beginning wrist calibration. Hit enter when done...");

    if (start_wrist_calibration(rightGlove)) {
        getchar();
        result = end_wrist_calibration(rightGlove);
    }

    return result;
}

bool calibrate_fingers(intptr_t rightGlove)
{
    bool result = false;

    printf("Beginning proximal calibration. Hit enter when done...");

    if (start_proximal_calibration(rightGlove)) {
        getchar();
        result = end_proximal_calibration(rightGlove);
    }

    return result;
}

bool calibrate_thumb(intptr_t rightGlove)
{
    bool result = false;

    printf("Beginning distal calibration. Hit enter when done...");
```

```
    if (start_distal_calibration(rightGlove)) {
        getchar();
        result = end_distal_calibration(rightGlove);
    }

    return result;
}

int main(int argc, char *argv[])
{
    int exit_status = EXIT_SUCCESS;

    //Start the Maestro detection thread. This thread will handle connecting/disconnecting Maestro gloves
    if (start_maestro_detection_service()) {
        intptr_t rightGlove = get_right_glove_pointer();

        printf("Waiting for right glove...\n");

        //Wait until right glove is connected
        while (!is_glove_connected(rightGlove)) {
            Sleep(250);
        }

        printf("Right glove found.\n");

        //Calibrate the Maestro
        if (calibrate_wrist(rightGlove)) {
            if (calibrate_fingers(rightGlove)) {
                if (calibrate_thumb(rightGlove)) {

                    printf("Maestro calibration completed!\n");

                    //Output wrist rotation
                    for (;;) {
                        printf("Wrist rotation: %f\n",
      get_wrist_proximal_rotation(rightGlove));
                        Sleep(100);
                    }
                }
            }
        }
    } else {
        fprintf(stderr, "Failed to start Maestro detection thread.\n");
        exit_status = EXIT_FAILURE;
    }

    return exit_status;
}
```

Calibration

The calibration process consists of three steps: calibrating the wrist, and then the proximal joints of the fingers and thumb, and then the thumb. **The alpha release of the Maestro does not include independent distal joint tracking.**

1. Calibrating the wrist: move your wrist through a full range of motion up and down. Try to keep your fingers flat (straight out) during this motion.

2. Calibrating the fingers: move your fingers through a full range of motion from flat to curled inward like a fist.

3. Calibrating the thumb: move your thumb through a full range of circular motion.

You can repeat each of these motions during their respective calibration step until satisfied with the resulting calibration. You do not need to move at the same speed as the examples shown; as long as you don't move too quickly and/or inconsistently the quality of the calibration should not be noticeably impacted.

Troubleshooting and debugging

Logging

The Maestro API generates log files to help aid debugging in the event that glove is not properly connecting.

The log files for the Maestro can be found in `%LOCALAPPDATA%\Contact Control Interfaces\Maestro\[VERSION]\Logs`. Inside the `Logs` folder there will be log files of the format `maestro-[TIMESTAMP].log`, where `[TIMESTAMP]` is the local date on which that particular log file was generated.

The log messages themselves are formatted as follows: `[LONG TIMESTAMP][CATEGORY]: [MESSAGE]`. The category is included to aid readability, and usually the `ERROR` category will prove to be most useful in debugging. The category `ERROR` will also log the last error from Windows API. A complete list of all possible Windows system error codes can be found here. Should the glove be successfully detected but fail to start, the log files will return one of the following error codes. Note that errors 5-10 deal with the motion ranges file specifically:

```
0 - Success (should never be logged, as this is not an error)
1 - Glove failed to start because it was already running.
2 - Failed to connect to the glove, usually because it was not plugged in.
3 - Failed to create a new thread for the connected glove.
4 - Deprecated
5 - Failed to find the appropriate motion range file, i.e. 'left_maestro_rotation_ranges.txt' or 'right_maestr
6 - A required rotation range entry is missing. For example, the 'IndexProximal' line is missing.
7 - A required rotation range entry is invalid. For example, having an extra comma or value.
8 - A rotation range entry is invalid, for instance having non-numeric characters in the rotation values.
9 - A rotation range is valid, but the values are in the incorrect order, that is to say not ascending.
10 - An entry's name is invalid, for instance having 'IndexPrimoxal' instead of 'IndexProximal'. Usually indic
```

These files are generated purely to help debug easy to fix issues with the system or configuration, and can be deleted without issue if need be. Please note that these logs may not encompass any error that can occur, but should help diagnose most errors. Also please note that these logs report only on the Maestro device itself, and will give no insight into your calibration, or any project-related configuration that may be affecting your development.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

Module Documentation

4.1 Maestro calibration functions

These functions are used to control the calibration of the Maestro glove.

Functions

- bool start_wrist_calibration (intptr_t maestroPtr)
- bool end_wrist_calibration (intptr_t maestroPtr)
- bool start_proximal_calibration (intptr_t maestroPtr)
- bool end_proximal_calibration (intptr_t maestroPtr)
- bool start_thumb_calibration (intptr_t maestroPtr)
- bool end_thumb_calibration (intptr_t maestroPtr)
- const CalibrationState *const get_calibration_state (intptr_t maestroPtr)

4.1.1 Detailed Description

These functions are used to control the calibration of the Maestro glove.

```
#include "maestro.h"
```

The Maestro should be calibrated "top down" in the order of Wrist, Proximal joints, and Distal joints. Each section can be recalibrated independently, but any sections "below" must also be recalibrated. For example, if you recalibrate the proximal joints, the wrist calibration will remain valid, but the distal joints will need to be recalibrated. If you recalibrate the wrist, both proximal and distal joints will need to be recalibrated.

4.1.2 Function Documentation

4.1.2.1 end_proximal_calibration()

```
bool end_proximal_calibration (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Whether or not proximal calibration was stopped. If the proximal joints were not calibrating or `maestroPtr` is invalid, `false` is returned.

4.1.2.2 end_thumb_calibration()

```
bool end_thumb_calibration (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Whether or not thumb calibration was stopped. If the thumb was not calibrating or `maestroPtr` is invalid, `false` is returned.

4.1.2.3 end_wrist_calibration()

```
bool end_wrist_calibration (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Whether or not wrist calibration was stopped. If wrist was not calibrating or `maestroPtr` is invalid, `false` is returned.

4.1.2.4 get_calibration_state()

```
const CalibrationState* const get_calibration_state (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

A structure containing the current states for each calibration step.

4.1.2.5 start_proximal_calibration()

```
bool start_proximal_calibration (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Whether or not proximal calibration was started. If the proximal joints are already calibrating or `maestroPtr` is invalid, `false` is returned.

4.1.2.6 start_thumb_calibration()

```
bool start_thumb_calibration (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Whether or not thumb calibration was started. If the thumb was already calibrating or `maestroPtr` is invalid, `false` is returned.

4.1.2.7 start_wrist_calibration()

```
bool start_wrist_calibration (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Whether or not wrist calibration was started. If wrist is already calibrating or `maestroPtr` is invalid, `false` is returned.

4.2 Glove management functions

These functions are used to manage and reference the Maestro gloves.

Functions

- bool start_maestro_detection_service ()
- intptr_t const get_left_glove_pointer ()
- intptr_t const get_right_glove_pointer ()
- bool is_glove_connected (intptr_t maestroPtr)
- const MaestroGloveContext *const get_glove_context (intptr_t maestroPtr)

4.2.1 Detailed Description

These functions are used to manage and reference the Maestro gloves.

#include "maestro.h"

`start_maestro_detection_service()` must be called before calibration, haptics, or any motion capture data can be used. The returned pointers should be stored and passed into all other Maestro API functions.

4.2.2 Function Documentation

4.2.2.1 get_glove_context()

```
const MaestroGloveContext* const get_glove_context (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The current Maestro data. This returns a struct containing the same values acquired by calling `get_displacement_context()`, `get_vibration_context()`, `get_force_feedback_context()`, and `is_glove_connected()`.

4.2.2.2 get_left_glove_pointer()

```
intptr_t const get_left_glove_pointer ( )
```

Returns

The pointer to the left glove context. This value should be passed into other function that set or retrieve values for the left glove.

4.2.2.3 get_right_glove_pointer()

```
intptr_t const get_right_glove_pointer ( )
```

Returns

The pointer to the right glove context. This value should be passed into other functions that set or retrieve values for the right glove.

4.2.2.4 is_glove_connected()

```
bool is_glove_connected (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Whether or not the glove referenced by `maestroPtr` is connected.

4.2.2.5 start_maestro_detection_service()

```
bool start_maestro_detection_service ( )
```

Start the Maestro glove detection thread. This function must be called in order for the Maestro glove(s) to connect. This will spawn a new thread that manages the connecting and disconnecting of Maestro gloves.

Returns

Whether or not the detection thread started successfully.

4.3 Hand displacements access functions

These functions are used to get raw displacement data for specific hand and finger joints.

Functions

- float get_index_proximal_displacement (intptr_t maestroPtr)
- float get_index_distal_displacement (intptr_t maestroPtr)
- float get_middle_distal_displacement (intptr_t maestroPtr)
- float get_middle_proximal_displacement (intptr_t maestroPtr)
- float get_ring_distal_displacement (intptr_t maestroPtr)
- float get_ring_proximal_displacement (intptr_t maestroPtr)
- float get_little_distal_displacement (intptr_t maestroPtr)
- float get_little_proximal_displacement (intptr_t maestroPtr)
- float get_thumb_distal_displacement (intptr_t maestroPtr)
- float get_thumb_abduction_displacement (intptr_t maestroPtr)
- float get_thumb_proximal_displacement (intptr_t maestroPtr)
- float get_wrist_proximal_displacement (intptr_t maestroPtr)
- const DisplacementContext ∗const get_displacement_context (intptr_t maestroPtr)

4.3.1 Detailed Description

These functions are used to get raw displacement data for specific hand and finger joints.

```
#include "maestro.h"
```

The displacements will always be in the inclusive range of (0, 1023).

4.3.2 Function Documentation

4.3.2.1 get_displacement_context()

```
const DisplacementContext* const get_displacement_context (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current displacements from the hardware. This returns a struct containing the same values acquired by the individual `get_..._displacement()` functions.

4.3.2.2 get_index_distal_displacement()

```
float get_index_distal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the index finger distal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.3 get_index_proximal_displacement()

```
float get_index_proximal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the index finger proximal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.4 get_little_distal_displacement()

```
float get_little_distal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the little (pinky) finger distal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.5 get_little_proximal_displacement()

```
float get_little_proximal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the little (pinky) finger proximal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.6 get_middle_distal_displacement()

```
float get_middle_distal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the middle finger distal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.7 get_middle_proximal_displacement()

```
float get_middle_proximal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the middle finger proximal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.8 get_ring_distal_displacement()

```
float get_ring_distal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the ring finger distal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.9 get_ring_proximal_displacement()

```
float get_ring_proximal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the ring finger proximal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.10 get_thumb_abduction_displacement()

```
float get_thumb_abduction_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the thumb abduction joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.11 get_thumb_distal_displacement()

```
float get_thumb_distal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the thumb distal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.12 get_thumb_proximal_displacement()

```
float get_thumb_proximal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the thumb proximal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.3.2.13 get_wrist_proximal_displacement()

```
float get_wrist_proximal_displacement (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The displacement for the wrist proximal joint. This is the raw data from the hardware before calculating the corresponding rotation. The displacement will always be in the inclusive range of (0, 1023).

4.4 Hand rotations access functions

These functions are used to get the rotations for specific hand and finger joints.

Functions

- float get_index_proximal_rotation_ratio (intptr_t maestroPtr)
- float get_middle_proximal_rotation_ratio (intptr_t maestroPtr)
- float get_ring_proximal_rotation_ratio (intptr_t maestroPtr)
- float get_little_proximal_rotation_ratio (intptr_t maestroPtr)
- float get_thumb_proximal_rotation_ratio (intptr_t maestroPtr)
- float get_index_proximal_rotation (intptr_t maestroPtr)
- float get_index_distal_rotation (intptr_t maestroPtr)
- float get_middle_distal_rotation (intptr_t maestroPtr)
- float get_middle_proximal_rotation (intptr_t maestroPtr)
- float get_ring_distal_rotation (intptr_t maestroPtr)
- float get_ring_proximal_rotation (intptr_t maestroPtr)
- float get_little_distal_rotation (intptr_t maestroPtr)
- float get_little_proximal_rotation (intptr_t maestroPtr)
- float get_thumb_distal_rotation (intptr_t maestroPtr)
- float get_thumb_abduction_rotation (intptr_t maestroPtr)
- float get_thumb_proximal_rotation (intptr_t maestroPtr)
- float get_wrist_proximal_rotation (intptr_t maestroPtr)
- void get_rotation_context (intptr_t maestroPtr, RotationContext *pRotationContext)

4.4.1 Detailed Description

These functions are used to get the rotations for specific hand and finger joints.

```
#include "maestro.h"
```

4.4.2 Function Documentation

4.4.2.1 get_index_distal_rotation()

```
float get_index_distal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the index finger distal joint in degrees.

4.4.2.2 get_index_proximal_rotation()

```
float get_index_proximal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the index finger proximal joint in degrees.

4.4.2.3 get_index_proximal_rotation_ratio()

```
float get_index_proximal_rotation_ratio (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Returns a floating-point number in the inclusive range `(0, 1)` representing the current displacement in proportion to the displacement range. If the index finger proximal joint is at the maximum displacement (rotated all the way down), `1` is returned. If at the minimum displacement (all the way up, such as when the hand is held flat), `0` is returned. If between the minimum and maximum, the proportion is returned. For example, half way through the displacement range (halfway between the minimum and maximum rotations/displacements), `0.5` is returned.

4.4.2.4 get_little_distal_rotation()

```
float get_little_distal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the little (pinky) finger distal joint in degrees.

4.4.2.5 get_little_proximal_rotation()

```
float get_little_proximal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the little (pinky) finger proximal joint in degrees.

4.4.2.6 get_little_proximal_rotation_ratio()

```
float get_little_proximal_rotation_ratio (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Returns a floating-point number in the inclusive range `(0, 1)` representing the current displacement in proportion to the displacement range. If the little (pinky) finger proximal joint is at the maximum displacement (rotated all the way down), `1` is returned. If at the minimum displacement (all the way up, such as when the hand is held flat), `0` is returned. If between the minimum and maximum, the proportion is returned. For example, half way through the displacement range (halfway between the minimum and maximum rotations/displacements), `0.5` is returned.

4.4.2.7 get_middle_distal_rotation()

```
float get_middle_distal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the middle finger distal joint in degrees.

4.4.2.8 get_middle_proximal_rotation()

```
float get_middle_proximal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the middle finger proximal joint in degrees.

4.4.2.9 get_middle_proximal_rotation_ratio()

```
float get_middle_proximal_rotation_ratio (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Returns a floating-point number in the inclusive range `(0, 1)` representing the current displacement in proportion to the displacement range. If the middle finger proximal joint is at the maximum displacement (rotated all the way down), `1` is returned. If at the minimum displacement (all the way up, such as when the hand is held flat), `0` is returned. If between the minimum and maximum, the proportion is returned. For example, half

way through the displacement range (halfway between the minimum and maximum rotations/displacements), `0.5` is returned.

4.4.2.10 get_ring_distal_rotation()

```
float get_ring_distal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the ring finger distal joint in degrees.

4.4.2.11 get_ring_proximal_rotation()

```
float get_ring_proximal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the ring finger proximal joint in degrees.

4.4.2.12 get_ring_proximal_rotation_ratio()

```
float get_ring_proximal_rotation_ratio (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Returns a floating-point number in the inclusive range `(0, 1)` representing the current displacement in proportion to the displacement range. If the ring finger proximal joint is at the maximum displacement (rotated all the way down), `1` is returned. If at the minimum displacement (all the way up, such as when the hand is held flat), `0` is returned. If between the minimum and maximum, the proportion is returned. For example, half way through the displacement range (halfway between the minimum and maximum rotations/displacements), `0.5` is returned.

4.4.2.13 get_rotation_context()

```
void get_rotation_context (
            intptr_t maestroPtr,
            RotationContext * pRotationContext )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
pRotationContext	A pointer to a `RotationContext` to be updated with the current rotation values. These are the same values obtained by calling the `get_..._rotation()` functions independently.

4.4.2.14 get_thumb_abduction_rotation()

```
float get_thumb_abduction_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the thumb abduction joint in degrees.

4.4.2.15 get_thumb_distal_rotation()

```
float get_thumb_distal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the thumb distal joint in degrees.

4.4.2.16 get_thumb_proximal_rotation()

```
float get_thumb_proximal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the thumb proximal joint in degrees.

4.4.2.17 get_thumb_proximal_rotation_ratio()

```
float get_thumb_proximal_rotation_ratio (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Returns a floating-point number in the inclusive range `(0, 1)` representing the current displacement in proportion to the displacement range. If the thumb proximal joint is at the maximum displacement (rotated all the way down), `1` is returned. If at the minimum displacement (all the way up, such as when the hand is held flat), `0` is returned. If between the minimum and maximum, the proportion is returned. For example, half way through the displacement range (halfway between the minimum and maximum rotations/displacements), `0.5` is returned.

4.4.2.18 get_wrist_proximal_rotation()

```
float get_wrist_proximal_rotation (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

The rotation for the wrist proximal joint in degrees.

4.5 Haptic control functions

These functions are used to control the vibration and force-feedback haptics for an individual finger.

Modules

- Vibration control functions

  These functions are used to set the vibration effect for an individual finger.
- Force-feedback control functions

  These functions are used to set the force-feedback motor amplitude for an individual finger.

4.5.1 Detailed Description

These functions are used to control the vibration and force-feedback haptics for an individual finger.

4.6 Vibration control functions

These functions are used to set the vibration effect for an individual finger.

Functions

- void set_thumb_vibration_effect (intptr_t maestroPtr, uint8_t effect)
- void set_index_vibration_effect (intptr_t maestroPtr, uint8_t effect)
- void set_middle_vibration_effect (intptr_t maestroPtr, uint8_t effect)
- void set_ring_vibration_effect (intptr_t maestroPtr, uint8_t effect)
- void set_little_vibration_effect (intptr_t maestroPtr, uint8_t effect)
- uint8_t get_thumb_vibration_effect (intptr_t maestroPtr)
- uint8_t get_index_vibration_effect (intptr_t maestroPtr)
- uint8_t get_middle_vibration_effect (intptr_t maestroPtr)
- uint8_t get_ring_vibration_effect (intptr_t maestroPtr)
- uint8_t get_little_vibration_effect (intptr_t maestroPtr)
- const VibrationContext *const get_vibration_context (intptr_t maestroPtr)

4.6.1 Detailed Description

These functions are used to set the vibration effect for an individual finger.

```
#include "maestro.h"
```

```
Here's a good article if you're unsure which effect you should use for certain
interactions.
```

Below is a list of available vibration effects (the DRV2605 haptic driver is used):

4.6.2 Function Documentation

4.6.2.1 get_index_vibration_effect()

```
uint8_t get_index_vibration_effect (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current vibration effect for the index finger.

4.6.2.2 get_little_vibration_effect()

```
uint8_t get_little_vibration_effect (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current vibration effect for the little (pinky) finger.

4.6.2.3 get_middle_vibration_effect()

```
uint8_t get_middle_vibration_effect (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current vibration effect for the middle finger.

4.6.2.4 get_ring_vibration_effect()

```
uint8_t get_ring_vibration_effect (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current vibration effect for the ring finger.

4.6.2.5 get_thumb_vibration_effect()

```
uint8_t get_thumb_vibration_effect (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current vibration effect for the thumb.

4.6.2.6 get_vibration_context()

```
const VibrationContext* const get_vibration_context (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current vibration effects. This returns a struct containing the same values acquired by `get_thumb_vibration_effect()`, `get_index_vibration_effect()`, `get_middle_vibration_effect()`, `get_ring_vibration_effect()`, and `get_little_vibration_effect()`

4.6.2.7 set_index_vibration_effect()

```
void set_index_vibration_effect (
            intptr_t maestroPtr,
            uint8_t effect )
```

Sets the vibration effect for the index finger.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
effect	The vibration effect to be used, or 0 for no vibration.

4.6.2.8 set_little_vibration_effect()

```
void set_little_vibration_effect (
            intptr_t maestroPtr,
            uint8_t effect )
```

Sets the vibration effect for the little (pinky) finger.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
effect	The vibration effect to be used, or 0 for no vibration.

4.6.2.9 set_middle_vibration_effect()

```
void set_middle_vibration_effect (
            intptr_t maestroPtr,
            uint8_t effect )
```

Sets the vibration effect for the middle finger.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
effect	The vibration effect to be used, or 0 for no vibration.

4.6.2.10 set_ring_vibration_effect()

```
void set_ring_vibration_effect (
            intptr_t maestroPtr,
            uint8_t effect )
```

Sets the vibration effect for the ring finger.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
effect	The vibration effect to be used, or 0 for no vibration.

4.6.2.11 set_thumb_vibration_effect()

```
void set_thumb_vibration_effect (
            intptr_t maestroPtr,
            uint8_t effect )
```

Sets the vibration effect for the thumb.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
effect	The vibration effect to be used, or 0 for no vibration.

4.7 Force-feedback control functions

These functions are used to set the force-feedback motor amplitude for an individual finger.

Functions

- void set_thumb_motor_amplitude (intptr_t maestroPtr, uint8_t amplitude)
- void set_index_motor_amplitude (intptr_t maestroPtr, uint8_t amplitude)
- void set_middle_motor_amplitude (intptr_t maestroPtr, uint8_t amplitude)
- void set_ring_motor_amplitude (intptr_t maestroPtr, uint8_t amplitude)
- void set_little_motor_amplitude (intptr_t maestroPtr, uint8_t amplitude)
- uint8_t get_thumb_motor_amplitude (intptr_t maestroPtr)
- uint8_t get_index_motor_amplitude (intptr_t maestroPtr)
- uint8_t get_middle_motor_amplitude (intptr_t maestroPtr)
- uint8_t get_ring_motor_amplitude (intptr_t maestroPtr)
- uint8_t get_little_motor_amplitude (intptr_t maestroPtr)
- const ForceFeedbackContext *const get_force_feedback_context (intptr_t maestroPtr)
- uint8_t get_max_motor_amplitude ()
- uint8_t get_default_motor_amplitude ()

4.7.1 Detailed Description

These functions are used to set the force-feedback motor amplitude for an individual finger.

```
#include "maestro.h"
```

4.7.2 Function Documentation

4.7.2.1 get_default_motor_amplitude()

```
uint8_t get_default_motor_amplitude ( )
```

Returns

The default non-collision motor amplitude. It is recommended to have a motor amplitude of at least the default for the sake of avoiding any slack in the force-feedback tendons.

4.7.2.2 get_force_feedback_context()

```
const ForceFeedbackContext* const get_force_feedback_context (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current force-feedback motor amplitudes. This returns a struct containing the same values acquired by `get_thumb_motor_amplitude()`, `get_index_motor_amplitude()`, `get_middle_motor_amplitude()`, `get_ring_motor_amplitude()`, and `get_little_motor_amplitude()`

4.7.2.3 get_index_motor_amplitude()

```
uint8_t get_index_motor_amplitude (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current force-feedback motor amplitude for the index finger.

4.7.2.4 get_little_motor_amplitude()

```
uint8_t get_little_motor_amplitude (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current force-feedback motor amplitude for the little (pinky) finger.

4.7.2.5 get_max_motor_amplitude()

```
uint8_t get_max_motor_amplitude ( )
```

Returns

The maximum motor amplitude allowed.

4.7.2.6 get_middle_motor_amplitude()

```
uint8_t get_middle_motor_amplitude (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current force-feedback motor amplitude for the middle finger.

4.7.2.7 get_ring_motor_amplitude()

```
uint8_t get_ring_motor_amplitude (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current force-feedback motor amplitude for the ring finger.

4.7.2.8 get_thumb_motor_amplitude()

```
uint8_t get_thumb_motor_amplitude (
            intptr_t maestroPtr )
```

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().

Returns

Current force-feedback motor amplitude for the thumb.

4.7.2.9 set_index_motor_amplitude()

```
void set_index_motor_amplitude (
            intptr_t maestroPtr,
            uint8_t amplitude )
```

Sets the force-feedback motor amplitude for the index finger.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
amplitude	The new amplitude for the force-feedback motor.

4.7.2.10 set_little_motor_amplitude()

```
void set_little_motor_amplitude (
            intptr_t maestroPtr,
            uint8_t amplitude )
```

Sets the force-feedback motor amplitude for the little (pinky) finger.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
amplitude	The new amplitude for the force-feedback motor.

4.7.2.11 set_middle_motor_amplitude()

```
void set_middle_motor_amplitude (
            intptr_t maestroPtr,
            uint8_t amplitude )
```

Sets the force-feedback motor amplitude for the middle finger.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
amplitude	The new amplitude for the force-feedback motor.

4.7.2.12 set_ring_motor_amplitude()

```
void set_ring_motor_amplitude (
            intptr_t maestroPtr,
            uint8_t amplitude )
```

Sets the force-feedback motor amplitude for the ring finger.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
amplitude	The new amplitude for the force-feedback motor.

4.7.2.13 set_thumb_motor_amplitude()

```
void set_thumb_motor_amplitude (
            intptr_t maestroPtr,
            uint8_t amplitude )
```

Sets the force-feedback motor amplitude for the thumb.

Parameters

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or get_right_glove_pointer().
amplitude	The new amplitude for the force-feedback motor.

Chapter 5

Data Structure Documentation

5.1 CalibrationState Struct Reference

A structure containing the state of each calibration step.

```
#include "maestro_types.h"
```

Data Fields

- bool isCalibratingWrist

  Whether or not the wrist is calibrating.
- bool isCalibratingProximals

  Whether or not the proximal joints are calibrating.
- bool isCalibratingThumb

  Whether or not the thumb is calibrating.

5.1.1 Detailed Description

A structure containing the state of each calibration step.

5.2 DisplacementContext Struct Reference

A structure containing the raw displacements read from the Maestro hardware.

```
#include "maestro_types.h"
```

Data Fields

- float index_distal_displacement

  Displacement for the distal joint of the index finger.
- float index_proximal_displacement

  Displacement for the proximal joint of the index finger.
- float middle_distal_displacement

  Displacement for the distal joint of the middle finger.
- float middle_proximal_displacement

  Displacement for the proximal joint of the middle finger.
- float ring_distal_displacement

  Displacement for the distal joint of the ring finger.
- float ring_proximal_displacement

  Displacement for the proximal joint of the ring finger.
- float little_distal_displacement

  Displacement for the distal joint of the little finger.
- float little_proximal_displacement

  Displacement for the proximal joint of the little finger.
- float thumb_distal_displacement

  Displacement for the distal joint of the thumb.
- float thumb_proximal_displacement

  Displacement for the proximal joint of the thumb.
- float thumb_abduction_displacement

  Displacement for the abduction movement (side-to-side) of the proximal joint of the thumb.
- float wrist_proximal_displacement

  Displacement for the wrist.
- float wrist_abduction_displacement

  Displacement for the abduction (side-to-side) movement of the wrist.

5.2.1 Detailed Description

A structure containing the raw displacements read from the Maestro hardware.

These are the raw inputs that are used to calculate rotations of the fingers, thumb, and wrist. See `get_↩ displacement_context()`.

5.3 ForceFeedbackContext Struct Reference

A structure containing the force-feedback motor amplitudes for each finger.

```
#include "maestro_types.h"
```

Data Fields

- uint8_t thumb_motor_amplitude

  The force-feedback motor amplitude for the thumb.
- uint8_t index_motor_amplitude

  The force-feedback motor amplitude for the index finger.
- uint8_t middle_motor_amplitude

  The force-feedback motor amplitude for the middle finger.
- uint8_t ring_motor_amplitude

  The force-feedback motor amplitude for the ring finger.
- uint8_t little_motor_amplitude

  The force-feedback motor amplitude for the little finger.

5.3.1 Detailed Description

A structure containing the force-feedback motor amplitudes for each finger.

These are the amplitudes used to control the force-feedback motors for each finger. They are proportional to the power output by the motor, where `255` is the maximum force and `0` is no force. See `get_force_feedback_context()`.

5.4 MaestroGloveContext Struct Reference

A structure containing vibration, force feedback, raw displacement data, and whether the glove is connected.

```
#include "maestro_types.h"
```

Data Fields

- volatile DisplacementContext displacementContext

  The displacement context for the glove.
- volatile VibrationContext vibrationContext

  The vibration context for the glove.
- volatile ForceFeedbackContext forceFeedbackContext

  The force feedback for the glove.
- bool is_running

  Whether or not the glove is running. This is the same value returned by `is_glove_connected()`.

5.4.1 Detailed Description

A structure containing vibration, force feedback, raw displacement data, and whether the glove is connected.

See `DisplacementContext`, `VibrationContext`, and `VibrationContext`.

5.5 RotationContext Struct Reference

A structure containing the calculated rotations for each joint.

`#include "maestro_types.h"`

Data Fields

- float index_distal_rotation
  Rotation for the distal joint of the index finger.
- float index_proximal_rotation
  Rotation for the proximal joint of the index finger.
- float middle_distal_rotation
  Rotation for the distal joint of the middle finger.
- float middle_proximal_rotation
  Rotation for the proximal joint of the middle finger.
- float ring_distal_rotation
  Rotation for the distal joint of the ring finger.
- float ring_proximal_rotation
  Rotation for the proximal joint of the ring finger.
- float little_distal_rotation
  Rotation for the distal joint of the little finger.
- float little_proximal_rotation
  Rotation for the proximal joint of the little finger.
- float thumb_distal_rotation
  Rotation for the distal joint of the thumb.
- float thumb_proximal_rotation
  Rotation for the proximal joint of the thumb.
- float thumb_abduction_rotation
  Rotation for the abduction movement (side-to-side) of the proximal joint of the thumb.
- float wrist_proximal_rotation
  Rotation for the wrist.

5.5.1 Detailed Description

A structure containing the calculated rotations for each joint.

These rotations are calculated using the raw displacements (see `get_displacement_context()`) and the calibration.

5.6 VibrationContext Struct Reference

A structure containing the vibration effects for each finger.

`#include "maestro_types.h"`

Data Fields

- uint8_t thumb_vibration_effect

  The current vibration effect for the thumb.
- uint8_t index_vibration_effect

  The current vibration effect for the index finger.
- uint8_t middle_vibration_effect

  The current vibration effect for the middle finger.
- uint8_t ring_vibration_effect

  The current vibration effect for the ring finger.
- uint8_t little_vibration_effect

  The current vibration effect for the little finger.

5.6.1 Detailed Description

A structure containing the vibration effects for each finger.

These are the vibration effects associated with each finger. An effect of `0` means no vibration. A full list of vibration effects can be found on the main page of the documentation, and also on the page. See `get_vibration_context()`.

Index