# Maestro C API v0.1 Alpha

Generated by Doxygen 1.8.13

# **Contents**

1	Mair	n Page			1
2	Mod	ule Inde	ex		5
	2.1	Module	es		 5
3	Data	Struct	ure Index		7
	3.1	Data S	tructures		 7
4	Mod	ule Doc	umentati	on	9
	4.1	Maesti	o calibrati	ion functions	 9
		4.1.1	Detailed	Description	 9
		4.1.2	Function	Documentation	 9
			4.1.2.1	end_proximal_calibration()	 9
			4.1.2.2	end_thumb_calibration()	 10
			4.1.2.3	end_wrist_calibration()	 10
			4.1.2.4	get_calibration_state()	 11
			4.1.2.5	start_proximal_calibration()	 11
			4.1.2.6	start_thumb_calibration()	 11
			4.1.2.7	start_wrist_calibration()	 12
	4.2	Glove	managem	ent functions	 13
		4.2.1	Detailed	Description	 13
		4.2.2	Function	Documentation	 13
			4.2.2.1	get_glove_context()	 13
			4.2.2.2	get_left_glove_pointer()	 13
			4.2.2.3	get_right_glove_pointer()	 14

ii CONTENTS

		4.2.2.4	is_glove_connected()	14
		4.2.2.5	start_maestro_detection_service()	14
4.3	Hand o	displaceme	ents access functions	15
	4.3.1	Detailed	Description	15
	4.3.2	Function	Documentation	15
		4.3.2.1	get_displacement_context()	15
		4.3.2.2	get_index_distal_displacement()	16
		4.3.2.3	get_index_proximal_displacement()	16
		4.3.2.4	get_little_distal_displacement()	16
		4.3.2.5	get_little_proximal_displacement()	17
		4.3.2.6	get_middle_distal_displacement()	17
		4.3.2.7	get_middle_proximal_displacement()	17
		4.3.2.8	get_ring_distal_displacement()	18
		4.3.2.9	get_ring_proximal_displacement()	18
		4.3.2.10	get_thumb_abduction_displacement()	18
		4.3.2.11	get_thumb_distal_displacement()	19
		4.3.2.12	get_thumb_proximal_displacement()	19
		4.3.2.13	get_wrist_proximal_displacement()	20
4.4	Hand r	otations ac	ccess functions	21
	4.4.1	Detailed	Description	21
	4.4.2	Function	Documentation	21
		4.4.2.1	get_index_distal_rotation()	21
		4.4.2.2	get_index_proximal_rotation()	22
		4.4.2.3	get_index_proximal_rotation_ratio()	22
		4.4.2.4	get_little_distal_rotation()	22
		4.4.2.5	get_little_proximal_rotation()	23
		4.4.2.6	get_little_proximal_rotation_ratio()	23
		4.4.2.7	get_middle_distal_rotation()	24
		4.4.2.8	get_middle_proximal_rotation()	24
		4.4.2.9	get_middle_proximal_rotation_ratio()	24

CONTENTS

		4.4.2.10	get_ring_distal_rotation()	25
		4.4.2.11	get_ring_proximal_rotation()	25
		4.4.2.12	get_ring_proximal_rotation_ratio()	25
		4.4.2.13	get_rotation_context()	26
		4.4.2.14	get_thumb_abduction_rotation()	26
		4.4.2.15	get_thumb_distal_rotation()	26
		4.4.2.16	get_thumb_proximal_rotation()	27
		4.4.2.17	get_thumb_proximal_rotation_ratio()	27
		4.4.2.18	get_wrist_proximal_rotation()	28
4.5	Haptic	control fur	nctions	29
	4.5.1	Detailed	Description	29
4.6	Vibratio	on control	functions	30
	4.6.1	Detailed	Description	30
	4.6.2	Function	Documentation	30
		4.6.2.1	get_index_vibration_effect()	30
		4.6.2.2	get_little_vibration_effect()	31
		4.6.2.3	get_middle_vibration_effect()	31
		4.6.2.4	get_ring_vibration_effect()	31
		4.6.2.5	get_thumb_vibration_effect()	32
		4.6.2.6	get_vibration_context()	32
		4.6.2.7	set_index_vibration_effect()	32
		4.6.2.8	set_little_vibration_effect()	33
		4.6.2.9	set_middle_vibration_effect()	33
		4.6.2.10	set_ring_vibration_effect()	33
		4.6.2.11	set_thumb_vibration_effect()	34
4.7	Force-	feedback c	control functions	35
	4.7.1	Detailed	Description	35
	4.7.2	Function	Documentation	35
		4.7.2.1	get_default_motor_amplitude()	35
		4.7.2.2	get_force_feedback_context()	35
		4.7.2.3	get_index_motor_amplitude()	36
		4.7.2.4	get_little_motor_amplitude()	36
		4.7.2.5	get_max_motor_amplitude()	36
		4.7.2.6	get_middle_motor_amplitude()	37
		4.7.2.7	get_ring_motor_amplitude()	37
		4.7.2.8	get_thumb_motor_amplitude()	37
		4.7.2.9	set_index_motor_amplitude()	38
		4.7.2.10	set_little_motor_amplitude()	38
		4.7.2.11	set_middle_motor_amplitude()	38
		4.7.2.12	set_ring_motor_amplitude()	39
		4.7.2.13	set_thumb_motor_amplitude()	39

iv CONTENTS

5	Data Structure Documentation				
	5.1	Calibra	ttionState Struct Reference	41	
		5.1.1	Detailed Description	41	
	5.2	Displac	cementContext Struct Reference	41	
		5.2.1	Detailed Description	42	
	5.3	ForceF	eedbackContext Struct Reference	42	
		5.3.1	Detailed Description	43	
	5.4	Maestr	roGloveContext Struct Reference	43	
		5.4.1	Detailed Description	43	
	5.5	Rotatio	onContext Struct Reference	44	
		5.5.1	Detailed Description	44	
	5.6	Vibratio	onContext Struct Reference	44	
		5.6.1	Detailed Description	45	
Inc	dex			47	

# **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  $\DOCALAPPDATA\%\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 <math>\DOCALAPPDATA\%\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:

2 Main Page

```
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 <code>is\_glove\_connected(intptr\_t maestroPtr)</code> to determine if a glove is connected. Pass in the appropriate glove pointer obtained by <code>get\_left\_glove</code>—<code>pointer()</code> or <code>get\_right\_glove\_pointer()</code>.
- 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:

```
#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
                       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.

4 Main Page

# 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  $LOCALAPPDATA Contact Control Interfaces Maestro [V \leftarrow ERSION] Logs. Inside the Logs folder there will be log files of the format maestro-[TIMESTAMP].log, where [TIMESTAMP] is the local date on which that particular log file was generated.$ 

The log messages themselves are formatted as follows: <code>[LONG TIMESTAMP] [CATEGORY]: [MESSAGE]</code>. The category is included to aid readability, and usually the <code>ERROR</code> category will prove to be most useful in debugging. The category <code>ERROR</code> will also log the last error from Windows API. A complete <code>list</code> of all <code>possible Windows system error codes</code> 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_maestro_rotation_ranges.txt' or 'right_maestro_rotation_ranges.txt' or 'right_maestro_rotation_range entry is missing.
7 - A required rotation range entry is invalid. For example, the 'IndexProximal' line is missing.
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 indication.
```

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:

aestro calibration functions	9
love management functions	
and displacements access functions	15
and rotations access functions	21
aptic control functions	29
Vibration control functions	30
Force-feedback control functions	35

6 Module Index

# **Chapter 3**

# **Data Structure Index**

# 3.1 Data Structures

Here are the data structures with brief descriptions:

CalibrationState	
A structure containing the state of each calibration step	41
DisplacementContext	
A structure containing the raw displacements read from the Maestro hardware	41
ForceFeedbackContext	
A structure containing the force-feedback motor amplitudes for each finger	42
MaestroGloveContext	
A structure containing vibration, force feedback, raw displacement data, and whether the glove	
is connected	43
RotationContext	
A structure containing the calculated rotations for each joint	44
VibrationContext	
A structure containing the vibration effects for each finger	44

8 Data Structure Index

# **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()

# **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()

#### **Parameters**

ĺ	maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or	1
		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()

# **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()

# **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()

#### **Parameters**

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or	1
	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()

# Parameters

ſ	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()

# **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()

#### **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  $get \leftarrow \_displacement\_context()$ ,  $get\_vibration\_context()$ ,  $get\_force\_feedback\_ \leftarrow context()$ , 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()

#### **Parameters**

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or	1
	get_right_glove_pointer().	

# Returns

Whether or not the glove referenced by  ${\tt maestroPtr}\: is \: connected.$ 

# 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()

# **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()

#### **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()

# **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()

#### **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()

# **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()

#### **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()

# **Parameters**

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or	1
	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()

#### **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()

# 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()

#### **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**

ſ	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()

# **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()

#### **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()

#### **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()

# **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()

# **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()

#### **Parameters**

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or	1
	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()

# **Parameters**

Ī	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()

#### **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()

#### **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()

# **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()

# **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()

# Parameters

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

# Returns

The rotation for the ring finger proximal joint in degrees.

# 4.4.2.12 get\_ring\_proximal\_rotation\_ratio()

# **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()

#### **Parameters**

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

# 4.4.2.14 get\_thumb\_abduction\_rotation()

# **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()

# **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()

#### **Parameters**

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or	1
	get_right_glove_pointer().	

# Returns

The rotation for the thumb proximal joint in degrees.

# 4.4.2.17 get\_thumb\_proximal\_rotation\_ratio()

# 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()

# **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()

# **Parameters**

maestro	Ptr	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()

#### **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()

#### **Parameters**

1	naestroPtr	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()

#### **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()

#### **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()

#### **Parameters**

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or	1
	get_right_glove_pointer().	

#### Returns

Current vibration effects. This returns a struct containing the same values acquired by  $get\_thumb \leftarrow \_vibration\_effect()$ ,  $get\_index\_vibration\_effect()$ ,  $get\_middle\_vibration\_\leftrightarrow effect()$ ,  $get\_ring\_vibration\_effect()$ , and  $get\_little\_vibration\_effect()$ 

#### 4.6.2.7 set\_index\_vibration\_effect()

Sets the vibration effect for the index finger.

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()

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()

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()

Sets the vibration effect for the ring finger.

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()

Sets the vibration effect for the thumb.

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()

#### **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()

#### **Parameters**

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or	1
	get_right_glove_pointer().	

#### Returns

Current force-feedback motor amplitude for the index finger.

# 4.7.2.4 get\_little\_motor\_amplitude()

#### **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()

#### **Parameters**

maestroPtr 7

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()

### **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()

maestroPtr	The pointer the Maestro Glove context. Obtained by get_left_glove_pointer() or	7
	get_right_glove_pointer().	

#### Returns

Current force-feedback motor amplitude for the thumb.

# 4.7.2.9 set\_index\_motor\_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()

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()

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()

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()

Sets the force-feedback motor amplitude for the thumb.

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  $get\_ \leftarrow displacement\_context()$ .

#### 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  $get\_force\_feedback \leftarrow \_context()$ .

# 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  $get\_vibration\_\leftarrow context$  ().

# Index

CalibrationState, 41	Vibration control functions, 30
	get_left_glove_pointer
DisplacementContext, 41	Glove management functions, 13
	get_little_distal_displacement
end_proximal_calibration	Hand displacements access functions, 16
Maestro calibration functions, 9	get_little_distal_rotation
end_thumb_calibration	Hand rotations access functions, 22
Maestro calibration functions, 10	get_little_motor_amplitude
end_wrist_calibration	Force-feedback control functions, 36
Maestro calibration functions, 10	get_little_proximal_displacement
	Hand displacements access functions, 17
Force-feedback control functions, 35	get_little_proximal_rotation
get_default_motor_amplitude, 35	Hand rotations access functions, 23
get_force_feedback_context, 35	get_little_proximal_rotation_ratio
get_index_motor_amplitude, 36	Hand rotations access functions, 23
get_little_motor_amplitude, 36	get_little_vibration_effect
get_max_motor_amplitude, 36	Vibration control functions, 30
get_middle_motor_amplitude, 37	get_max_motor_amplitude
get_ring_motor_amplitude, 37	Force-feedback control functions, 36
get_thumb_motor_amplitude, 37	get_middle_distal_displacement
set_index_motor_amplitude, 38	Hand displacements access functions, 17
set_little_motor_amplitude, 38	get_middle_distal_rotation
set_middle_motor_amplitude, 38	Hand rotations access functions, 23
set_ring_motor_amplitude, 39	get_middle_motor_amplitude
set_thumb_motor_amplitude, 39	Force-feedback control functions, 37
ForceFeedbackContext, 42	get_middle_proximal_displacement
	Hand displacements access functions, 17
get_calibration_state	get_middle_proximal_rotation
Maestro calibration functions, 10	Hand rotations access functions, 24
get_default_motor_amplitude	get_middle_proximal_rotation_ratio
Force-feedback control functions, 35	Hand rotations access functions, 24
get_displacement_context	get_middle_vibration_effect
Hand displacements access functions, 15	Vibration control functions, 31
get_force_feedback_context	get_right_glove_pointer
Force-feedback control functions, 35	
get_glove_context	Glove management functions, 14
Glove management functions, 13	get_ring_distal_displacement
get_index_distal_displacement	Hand displacements access functions, 18
Hand displacements access functions, 15	get_ring_distal_rotation
get_index_distal_rotation	Hand rotations access functions, 25
Hand rotations access functions, 21	get_ring_motor_amplitude
get_index_motor_amplitude	Force-feedback control functions, 37
Force-feedback control functions, 36	get_ring_proximal_displacement
get_index_proximal_displacement	Hand displacements access functions, 18
Hand displacements access functions, 16	get_ring_proximal_rotation
get_index_proximal_rotation	Hand rotations access functions, 25
Hand rotations access functions, 22	get_ring_proximal_rotation_ratio
get_index_proximal_rotation_ratio	Hand rotations access functions, 25
Hand rotations access functions, 22	get_ring_vibration_effect
get index vibration effect	Vibration control functions 31

48 INDEX

get_rotation_context	get_ring_proximal_rotation_ratio, 25
Hand rotations access functions, 26	get_rotation_context, 26
get_thumb_abduction_displacement	get_thumb_abduction_rotation, 26
Hand displacements access functions, 18	get_thumb_distal_rotation, 26
get_thumb_abduction_rotation	<pre>get_thumb_proximal_rotation, 27</pre>
Hand rotations access functions, 26	get_thumb_proximal_rotation_ratio, 27
get_thumb_distal_displacement	get_wrist_proximal_rotation, 27
Hand displacements access functions, 19	Haptic control functions, 29
get_thumb_distal_rotation	
Hand rotations access functions, 26	is_glove_connected
get_thumb_motor_amplitude	Glove management functions, 14
Force-feedback control functions, 37	
get_thumb_proximal_displacement	Maestro calibration functions, 9
Hand displacements access functions, 19	end_proximal_calibration, 9
get_thumb_proximal_rotation	end_thumb_calibration, 10
Hand rotations access functions, 27	end_wrist_calibration, 10
get_thumb_proximal_rotation_ratio	get_calibration_state, 10
Hand rotations access functions, 27	start_proximal_calibration, 11
get thumb vibration effect	start_thumb_calibration, 11
Vibration control functions, 31	start_wrist_calibration, 11
get_vibration_context	MaestroGloveContext, 43
Vibration control functions, 32	
get_wrist_proximal_displacement	RotationContext, 44
Hand displacements access functions, 19	
get_wrist_proximal_rotation	set_index_motor_amplitude
Hand rotations access functions, 27	Force-feedback control functions, 38
Glove management functions, 13	set_index_vibration_effect
get_glove_context, 13	Vibration control functions, 32
get_left_glove_pointer, 13	set_little_motor_amplitude
get_right_glove_pointer, 14	Force-feedback control functions, 38
is_glove_connected, 14	set_little_vibration_effect
start_maestro_detection_service, 14	Vibration control functions, 33
Start_maestro_detection_service, 14	set_middle_motor_amplitude
Hand displacements access functions, 15	Force-feedback control functions, 38
get_displacement_context, 15	set_middle_vibration_effect
get_index_distal_displacement, 15	Vibration control functions, 33
get_index_proximal_displacement, 16	set_ring_motor_amplitude
get_little_distal_displacement, 16	Force-feedback control functions, 39
get_little_proximal_displacement, 17	set_ring_vibration_effect
get_middle_distal_displacement, 17	Vibration control functions, 33
get_middle_proximal_displacement, 17	set_thumb_motor_amplitude
get ring distal displacement, 18	Force-feedback control functions, 39
get_ring_distal_displacement, 18	set_thumb_vibration_effect
get_thumb_abduction_displacement, 18	Vibration control functions, 34
get_thumb_distal_displacement, 19	start maestro detection service
get_thumb_proximal_displacement, 19	Glove management functions, 14
	•
get_wrist_proximal_displacement, 19	start_proximal_calibration
Hand rotations access functions, 21	Maestro calibration functions, 11
get_index_distal_rotation, 21	start_thumb_calibration
get_index_proximal_rotation, 22	Maestro calibration functions, 11
get_index_proximal_rotation_ratio, 22	start_wrist_calibration
get_little_distal_rotation, 22	Maestro calibration functions, 11
get_little_proximal_rotation, 23	
get_little_proximal_rotation_ratio, 23	Vibration control functions, 30
get_middle_distal_rotation, 23	get_index_vibration_effect, 30
get_middle_proximal_rotation, 24	get_little_vibration_effect, 30
get_middle_proximal_rotation_ratio, 24	get_middle_vibration_effect, 31
get_ring_distal_rotation, 25	get_ring_vibration_effect, 31
get_ring_proximal_rotation, 25	get_thumb_vibration_effect, 31

INDEX 49

```
get_vibration_context, 32
set_index_vibration_effect, 32
set_little_vibration_effect, 33
set_middle_vibration_effect, 33
set_ring_vibration_effect, 33
set_thumb_vibration_effect, 34
VibrationContext, 44
```