Software documentation - API

Generated by Doxygen 1.8.13

Contents

Chapter 1

qbAPI Libraries

Those functions allows to use the board through a serial port

Version

7.0.0

Author

Mattia Poggiani

Date

January 25th, 2019

This is a set of functions that allows to use the board via a serial port.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 comm_settings Struct Reference

Data Fields

- HANDLE **file_handle**

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

- **qbmove_communications.h**

Chapter 5

File Documentation

5.1 commands.h File Reference

Definitions for board commands, parameters and packages.

This graph shows which files directly or indirectly include this file:

5.2 cp_commands.h File Reference

Definitions for additional commands, parameters and packages.

```
#include "commands.h"
```

Include dependency graph for cp_commands.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **API_VERSION** "v7.0.0 Centro Piaggio"
- #define **NUM_OF_ADDITIONAL_EMGS** 6
- #define **NUM_OF_INPUT_EMGS** 2

Enumerations

Additional Commands

- enum **additional_command** {
  CMD_GET_IMU_READINGS = 161, **CMD_GET_IMU_PARAM** = 162, **CMD_GET_ENCODER_CONF** = 163, **CMD_GET_ENCODER_RAW** = 164,
  CMD_GET_ADC_CONF = 165, **CMD_GET_ADC_RAW** = 166 }

5.2.1 Detailed Description

Definitions for additional commands, parameters and packages.

This file is included in the board firmware, in its libraries and applications. It contains all definitions that are necessary for the contruction of communication packages.

It includes definitions for all of the device commands, parameters and also the size of answer packages.

5.2.2 Macro Definition Documentation

5.2.2.1 NUM_OF_ADDITIONAL_EMGS

```
#define NUM_OF_ADDITIONAL_EMGS 6
```

Number of additional emg channels.

5.3 cp_communications.cpp File Reference

Library of functions for serial port communication with a board in addition to standard qbmove_communications.

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <ctype.h>
#include <time.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
#include <sys/ioctl.h>
#include <dirent.h>
#include <sys/time.h>
#include <stdlib.h>
#include <linux/serial.h>
#include "cp_communications.h"
#include "cp_commands.h"
```

Include dependency graph for cp_communications.cpp:

Macros

- #define **BUFFER_SIZE** 500

  Size of buffers that store communication packets.

5.3.1 Detailed Description

Library of functions for serial port communication with a board in addition to standard qbmove_communications.

Check the **cp_communications.h** (p. 10) file for a complete description of the public functions implemented in **cp_communications.cpp** (p. 10).

5.4 cp_communications.h File Reference

Library of functions for SERIAL PORT communication additional to standard qbmove_communications. Function Prototypes.

```
#include "qbmove_communications.h"
```

Include dependency graph for cp_communications.h: This graph shows which files directly or indirectly include this file:

Functions

- int **commGetImuReadings** (**comm_settings** ∗comm_settings_t, int id, uint8_t ∗imu_table, uint8_t ∗imus↩ _magcal, int n_imu, float ∗imu_values)
- int **commGetIMUParamList** (**comm_settings** ∗comm_settings_t, int id, unsigned short index, void ∗values, unsigned short value_size, unsigned short num_of_values, uint8_t ∗buffer)
- int **commGetEncoderConf** (**comm_settings** ∗comm_settings_t, int id, uint8_t ∗num_enc_line, uint8_↩ t ∗num_enc_per_line, uint8_t ∗enc_map)
- int **commGetEncoderRawValues** (**comm_settings** ∗comm_settings_t, int id, uint8_t enc_total, uint16_t enc_val[ ])
- int **commGetADCConf** (**comm_settings** ∗comm_settings_t, int id, uint8_t ∗num_ch, uint8_t ∗adc_map)
- int **commGetADCRawValues** (**comm_settings** ∗comm_settings_t, int id, uint8_t num_channels, short int adc[3])

5.4.1 Detailed Description

Library of functions for SERIAL PORT communication additional to standard qbmove_communications. Function Prototypes.

This library contains all necessary functions for communicating with a board when using a USB to RS485 connector that provides a Virtual COM interface.

5.4.2 Function Documentation

5.4.2.1 commGetADCConf()

```
int commGetADCConf (
              comm_settings * comm_settings_t,
             int id,
             uint8_t * num_ch,
             uint8_t * adc_map )
```

This function gets ADC map from board connected to the serial port.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(*p. 7*)* structure containing info about the communication settings.
id	The device's id number.

Returns

num_ch Number of channels.
adc_map Vector of adc map.

5.4.2.2 commGetADCRawValues()

```
int commGetADCRawValues (
                comm_settings * comm_settings_t,
               int id,
               uint8_t num_channels,
               short int adc[3] )
```

This function gets position measurements from a board connected to the serial port.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
num_channels	Number of channels.
adc	ADC raw values.

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
short int       adc[3];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetADCRaw(&comm_settings_t, DEVICE_ID, adc))
    printf("ADC raw: %d\t%d\t%d\n",adc[0], adc[1], adc[2]);
else
    puts("Couldn't retrieve measurements.");

closeRS485(&comm_settings_t);
```

5.4.2.3 commGetEncoderConf()

```
int commGetEncoderConf (
                comm_settings * comm_settings_t,
               int id,
               uint8_t * num_enc_line,
               uint8_t * num_enc_per_line,
               uint8_t * enc_map )
```

This function gets encoder map from board connected to the serial port.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
num_enc_line	Number of encoder lines.
num_enc_per_line	Number of encoder per line.

Returns

encoder_map Vector of encoder map.

5.4.2.4 commGetEncoderRawValues()

```
int commGetEncoderRawValues (
              comm_settings * comm_settings_t,
             int id,
             uint8_t enc_total,
             uint16_t enc_val[] )
```

This function gets encoder raw readings from board connected to the serial port.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.

Returns

encoder_val Vector of encoder map.

5.4.2.5 commGetIMUParamList()

```
int commGetIMUParamList (
              comm_settings * comm_settings_t,
             int id,
             unsigned short index,
             void * values,
             unsigned short value_size,
             unsigned short num_of_values,
             uint8_t * buffer )
```

This function gets all the parameters that are stored in the board memory and sets one of them if requested

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
index	The index relative to the parameter to be get.
values	An array with the parameter values.
value_size	The byte size of the parameter to be get
num_of_values	The size of the array of the parameter to be get
buffer	The array where the parameters' values and descriptions are saved

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
unsigned char   aux_string[2000];
int             index = 0;
int             value_size = 0;
int             num_of_values = 0;

// Get parameters
commGetIMUParamList(&comm_settings_t, device_id, index, NULL, value_size, num_of_values, aux_string);
string_unpacking_and_printing(aux_string);

// Set parameters

float           val[5];
val[0] = 1;
val[1] = 1;
val[2] = 0;
val[3] = 0;
val[4] = 0;
index = 2;
value_size = 6;
num_of_values = 5;
commGetIMUParamList(&comm_settings_t, device_id, index, pid, value_size, num_of_values, NULL);
```

5.4.2.6 commGetImuReadings()

```
int commGetImuReadings (
              comm_settings * comm_settings_t,
             int id,
             uint8_t * imu_table,
             uint8_t * imus_magcal,
             int n_imu,
             float * imu_values )
```

This function gets IMU readings from IMU board connected to the serial port.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
imu_table	IMU table configuration.
imus_magcal	IMU magnetometer calibratino parameters vector.
n_imu	Number of connected IMUs.

Returns

imu_values Vector of imu readings.

5.5 qbmove_communications.cpp File Reference

Library of functions for serial port communication with a board.

```
#include <stdio.h>
#include <string.h>
```

```
#include <stdint.h>
#include <ctype.h>
#include <time.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
#include <sys/ioctl.h>
#include <dirent.h>
#include <sys/time.h>
#include <stdlib.h>
#include <linux/serial.h>
#include "qbmove_communications.h"
#include "commands.h"
```

Include dependency graph for qbmove_communications.cpp:

Macros

- #define **BUFFER_SIZE** 500

  Size of buffers that store communication packets.

5.5.1 Detailed Description

Library of functions for serial port communication with a board.

Date

May 03, 2018

Author

Centro "E.Piaggio"

Copyright

Check the **qbmove_communications.h** (p. 15) file for a complete description of the public functions implemented in **qbmove_communications.cpp** (p. 14).

5.6 qbmove_communications.h File Reference

Library of functions for SERIAL PORT communication with a board. Function Prototypes.

```
#include <termios.h>
#include "commands.h"
#include <stdint.h>
```

Include dependency graph for qbmove_communications.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct **comm_settings**

Macros

- #define **HANDLE** int
- #define **INVALID_HANDLE_VALUE** -1
- #define **BAUD_RATE_T_2000000** 0
- #define **BAUD_RATE_T_460800** 1
- #define **MAX_WATCHDOG_TIME** 500
- #define **READ_TIMEOUT** 4000

Typedefs

- typedef struct **comm_settings comm_settings**

Functions

Virtual COM (RS485) functions

- int **RS485listPorts** (char list_of_ports[60][255])
- void **openRS485** (**comm_settings** ∗comm_settings_t, const char ∗port_s, int BAUD_RATE=B2000000)
- void **closeRS485** (**comm_settings** ∗comm_settings_t)
- int **RS485read** (**comm_settings** ∗comm_settings_t, int id, char ∗package)
- int **RS485ListDevices** (**comm_settings** ∗comm_settings_t, char list_of_ids[255])
- void **RS485GetInfo** (**comm_settings** ∗comm_settings_t, char ∗buffer)

qbAPI Commands

- int **commPing** (**comm_settings** ∗comm_settings_t, int id)
- void **commActivate** (**comm_settings** ∗comm_settings_t, int id, char activate)
- void **commSetBaudRate** (**comm_settings** ∗comm_settings_t, int id, short int baudrate)
- void **commSetWatchDog** (**comm_settings** ∗comm_settings_t, int id, short int wdt)
- void **commSetInputs** (**comm_settings** ∗comm_settings_t, int id, short int inputs[ ])
- void **commSetPosStiff** (**comm_settings** ∗comm_settings_t, int id, short int inputs[ ])
- int **commGetInputs** (**comm_settings** ∗comm_settings_t, int id, short int inputs[2])
- int **commGetMeasurements** (**comm_settings** ∗comm_settings_t, int id, short int measurements[3])
- int **commGetCounters** (**comm_settings** ∗comm_settings_t, int id, short unsigned int counters[20])
- int **commGetCurrents** (**comm_settings** ∗comm_settings_t, int id, short int currents[2])
- int **commGetCurrAndMeas** (**comm_settings** ∗comm_settings_t, int id, short int ∗values)
- int **commGetEmg** (**comm_settings** ∗comm_settings_t, int id, short int emg[2])
- int **commGetVelocities** (**comm_settings** ∗comm_settings_t, int id, short int measurements[ ])
- int **commGetAccelerations** (**comm_settings** ∗comm_settings_t, int id, short int measurements[ ])
- int **commGetActivate** (**comm_settings** ∗comm_settings_t, int id, char ∗activate)
- int **commGetInfo** (**comm_settings** ∗comm_settings_t, int id, short int info_type, char ∗info)
- int **commBootloader** (**comm_settings** ∗comm_settings_t, int id)
- int **commCalibrate** (**comm_settings** ∗comm_settings_t, int id)
- int **commHandCalibrate** (**comm_settings** ∗comm_settings_t, int id, short int speed, short int repetitions)

qbAPI Parameters

- int **commSetZeros** (**comm_settings** ∗comm_settings_t, int id, void ∗values, unsigned short num_of_↩ values)

- int **commGetParamList** (**comm_settings** ∗comm_settings_t, int id, unsigned short index, void ∗values, unsigned short value_size, unsigned short num_of_values, uint8_t ∗buffer)
- int **commStoreParams** (**comm_settings** ∗comm_settings_t, int id)
- int **commStoreDefaultParams** (**comm_settings** ∗comm_settings_t, int id)
- int **commRestoreParams** (**comm_settings** ∗comm_settings_t, int id)
- int **commInitMem** (**comm_settings** ∗comm_settings_t, int id)

General Functions

- long **timevaldiff** (struct timeval ∗starttime, struct timeval ∗finishtime)
- char **checksum** (char ∗data_buffer, int data_length)

Functions for other devices

- int **commExtDrive** (**comm_settings** ∗comm_settings_t, int id, char ext_input)
- void **commSetCuffInputs** (**comm_settings** ∗comm_settings_t, int id, int flag)
- int **commGetJoystick** (**comm_settings** ∗comm_settings_t, int id, short int joystick[2])

5.6.1 Detailed Description

Library of functions for SERIAL PORT communication with a board. Function Prototypes.

Date

May 03, 2018

Author

Centro "E.Piaggio"

Copyright

This library contains all necessary functions for communicating with a board when using a USB to RS485 connector that provides a Virtual COM interface.

5.6.2 Function Documentation

5.6.2.1 checksum()

```
char checksum (
            char * data_buffer,
            int data_length )
```

This functions returns an 8 bit LCR checksum over the lenght of a buffer.

Parameters

data_buffer	Buffer.
data_length	Buffer length.

Example

```
char    aux;
char    buffer[5];

buffer  = "abcde";
aux     = checksum(buffer,5);
printf("Checksum: %d", (int) aux)
```

5.6.2.2 closeRS485()

```
void closeRS485 (
              comm_settings * comm_settings_t )
```

This function is used to close a serial port being used with the board.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.

Example

```
comm_settings   comm_settings_t;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
closeRS485(&comm_settings_t);
```

5.6.2.3 commActivate()

```
void commActivate (
              comm_settings * comm_settings_t,
             int id,
             char activate )
```

This function activates or deactivates a board connected to the serial port.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
activate	TRUE to turn motors on. FALSE to turn motors off.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
commActivate(&comm_settings_t, device_id, TRUE);
closeRS485(&comm_settings_t);
```

5.6.2.4 commBootloader()

```
int commBootloader (
              comm_settings * comm_settings_t,
             int id )
```

This function sends the board in bootloader modality in order to update the firmware on the board

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.

Returns

Return 0 on success, -1 otherwise

Example

```
comm_settings comm_settings_t;
int     device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
commBootloader(&comm_settings_t, device_id);
closeRS485(&comm_settings_t);
```

5.6.2.5 commCalibrate()

```
int commCalibrate (
              comm_settings * comm_settings_t,
             int id )
```

This function is used to calibrate the maximum stiffness value of the board

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.

Returns

Returns 0 on success, -1 otherwise

Example

```
comm_settings comm_settings_t;
int     device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
commCalibrate(&comm_settings_t, device_id);
closeRS485(&comm_settings_t);
```

5.6.2.6 commExtDrive()

```
int commExtDrive (
              comm_settings * comm_settings_t,
             int id,
             char ext_input )
```

This function is used with the armslider device. Is used to drive another board with the inputs of the first one

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the comunication settings.
id	The id of the board drive.
ext_input	A flag used to activate the external drive functionality of the board.

Returns

A negative value if something went wrong, a zero if everything went fine.

5.6.2.7 commGetAccelerations()

```
int commGetAccelerations (
              comm_settings * comm_settings_t,
             int id,
             short int measurements[] )
```

This function gets the acceleration of the qbHand motor

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
measurements	Velocity measurements.

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings    comm_settings_t;
int              device_id = 65;
short int        acc_measurements[3];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetAccelerations(&comm_settings_t, device_id, acc_measurements))
    printf("Measurements: %d\t%d\t%d\n", acc_measurements[0], acc_measurements[1], acc_measurements[2]);
else
    puts("Couldn't retrieve accelerations.");

closeRS485(&comm_settings_t);
```

5.6.2.8 commGetActivate()

```
int commGetActivate (
              comm_settings * comm_settings_t,
             int id,
             char * activate )
```

This function gets the activation status of a board connected to the serial port.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
activation	Activation status.

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings comm_settings_t;
int     device_id = 65;
char    activation_status;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetActivate(&comm_settings_t, DEVICE_ID, activation_status))
    printf("Activation status: %d\n", &activation_status);
else
    puts("Couldn't retrieve activation status.");

closeRS485(&comm_settings_t);
```

5.6.2.9 commGetCounters()

```
int commGetCounters (
              comm_settings * comm_settings_t,
             int id,
             short unsigned int counters[20] )
```

This function gets counters values from a board connected to the serial port.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
counters	Counters

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings       comm_settings_t;
int                 device_id = 65;
short unsigned int  counters[20];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetCounters(&comm_settings_t, DEVICE_ID, counters))
    printf("Counters: %d\t%d\t {...} %d\n", counters[0], counters[1], {...}, counters[20]);
else
    puts("Couldn't retrieve counters.");

closeRS485(&comm_settings_t);
```

5.6.2.10 commGetCurrAndMeas()

```
int commGetCurrAndMeas (
              comm_settings * comm_settings_t,
             int id,
             short int * values )
```

This function gets currents and position measurements from a board connected to the serial port

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
values	Current and position measurements. Currents are in first two positions

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings    comm_settings_t;
int              device_id = 65;
short int        values[5];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

 if(!commGetCurrAndMeas(&comm_settings_t, device_id, currents)){
     printf("Currents: %d\t%d\t%d\n",values[0], values[1]);
     printf("Measurements: %d\t%d\t%d\n", values[2], values[3], values[4]);
 }
 else
     puts("Couldn't retrieve currents.");

closeRS485(&comm_settings_t);
```

5.6.2.11 commGetCurrents()

```
int commGetCurrents (
              comm_settings * comm_settings_t,
             int id,
             short int currents[2] )
```

This function gets currents from a board connected to the serial port.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
currents	Currents.

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings    comm_settings_t;
int              device_id = 65;
short int        currents[2];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetCurrents(&comm_settings_t, device_id, currents))
    printf("Measurements: %d\t%d\t%d\n",currents[0], currents[1]);
else
    puts("Couldn't retrieve currents.");

closeRS485(&comm_settings_t);
```

5.6.2.12 commGetEmg()

```
int commGetEmg (
                comm_settings * comm_settings_t,
               int id,
               short int emg[2] )
```

This function gets measurements from electomyographics sensors connected to the qbHand. IS USED ONLY W↩ HEN THE BOARD IS USED FOR A QBHAND

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
values	Emg sensors measurements.

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings    comm_settings_t;
int              device_id = 65;
short int        values[2];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetEmg(&comm_settings_t, device_id, values));
    printf("Measurements: %d\t%d\t%d\n", values[0], values[1]);
else
    puts("Couldn't retrieve emg values.");

closeRS485(&comm_settings_t);
```

5.6.2.13 commGetInfo()

```
int commGetInfo (
                comm_settings * comm_settings_t,
               int id,
               short int info_type,
               char * info )
```

This function is used to ping the board and get information about the device.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
buffer	Buffer that stores a string with information about the device. BUFFER SIZE MUST BE AT LEAST 500.
info_type	Information to be retrieved.

Example

```
comm_settings comm_settings_t;
char    auxstring[500];
int     device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
commGetInfo(&comm_settings_t, device_id, INFO_ALL, auxstring);
puts(auxstring);
closeRS485(&comm_settings_t);
```

5.6.2.14 commGetInputs()

```
int commGetInputs (
            comm_settings * comm_settings_t,
            int id,
            short int inputs[2] )
```

This function gets input references from a board connected to the serial port.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
inputs	Input references.

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
short int       inputs[2];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetInputs(&comm_settings_t, DEVICE_ID, inputs))
    printf("Inputs: %d\t%d\n",inputs[0], inputs[1]);
else
    puts("Couldn't retrieve device inputs.");

closeRS485(&comm_settings_t);
```

5.6.2.15 commGetJoystick()

```
int commGetJoystick (
            comm_settings * comm_settings_t,
            int id,
            short int joystick[2] )
```

This function gets joystick measurementes from a softhand connected to the serial port.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
joystick	Joystick analog measurements.

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings    comm_settings_t;
int              device_id = 65;
short int        joystick[2];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetJoystick(&comm_settings_t, device_id, joystick))
    printf("Measurements: %d\t%d\t%d\n",joystick[0], joystick[1]);
else
    puts("Couldn't retrieve joystick measurements.");

closeRS485(&comm_settings_t);
```

5.6.2.16 commGetMeasurements()

```
int commGetMeasurements (
             comm_settings * comm_settings_t,
            int id,
            short int measurements[3] )
```

This function gets position measurements from a board connected to the serial port.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
measurements	Measurements.

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
short int       measurements[3];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetMeasurements(&comm_settings_t, DEVICE_ID, measurements))
    printf("Measurements: %d\t%d\t%d\n",measurements[0], measurements[1], measurements[2]);
```

```
    else
        puts("Couldn't retrieve measurements.");

    closeRS485(&comm_settings_t);
```

5.6.2.17 commGetParamList()

```
int commGetParamList (
                comm_settings * comm_settings_t,
                int id,
                unsigned short index,
                void * values,
                unsigned short value_size,
                unsigned short num_of_values,
                uint8_t * buffer )
```

This function gets all the parameters that are stored in the board memory and sets one of them if requested

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
index	The index relative to the parameter to be get.
values	An array with the parameter values.
value_size	The byte size of the parameter to be get
num_of_values	The size of the array of the parameter to be get
buffer	The array where the parameters' values and descriptions are saved

Example

```
    comm_settings   comm_settings_t;
    int             device_id = 65;
    unsigned char   aux_string[2000];
    int             index = 0;
    int             value_size = 0;
    int             num_of_values = 0;

    // Get parameters
    commGetParamList(&comm_settings_t, device_id, index, NULL, value_size, num_of_values, aux_string);
    string_unpacking_and_printing(aux_string);

    // Set parameters

    float           pid[3];
    pid[0] = 0.1;
    pid[1] = 0.2;
    pid[2] = 0.3;
    index = 2;
    value_size = 4;
    num_of_values = 3;
    commGetParamList(&comm_settings_t, device_id, index, pid, value_size, num_of_values, NULL);
```

5.6.2.18 commGetVelocities()

```
int commGetVelocities (
                comm_settings * comm_settings_t,
```

```
            int id,
            short int measurements[] )
```

This function gets velocities of the two motors and the shaft from a board connected to a serial port or from the only shaft of the qbHand

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
measurements	Velocity measurements.

Returns

Returns 0 if communication was ok, -1 otherwise.

Example

```
comm_settings    comm_settings_t;
int              device_id = 65;
short int        vel_measurements[3];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

if(!commGetVelocities(&comm_settings_t, device_id, vel_measurements))
    printf("Measurements: %d\t%d\t%d\n", vel_measurements[0], vel_measurements[1], vel_measurements[2]);
else
    puts("Couldn't retrieve velocities.");

closeRS485(&comm_settings_t);
```

5.6.2.19 commHandCalibrate()

```
int commHandCalibrate (
             comm_settings * comm_settings_t,
            int id,
            short int speed,
            short int repetitions )
```

This function is used to make a series of opening and closures of the qbHand

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
speed	The speed of hand closure and opening [0 - 200]
repetitions	The nnumber of closures needed to be done [0 - 32767]

Example

```
comm_settings comm_settings_t;
```

```
int     speed = 200
int     repetitions = 400;
int     device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
commHandCalibrate(&comm_settings_t, device_id, speed, repetitions);
closeRS485(&comm_settings_t);
```

5.6.2.20 commInitMem()

```
int commInitMem (
              comm_settings * comm_settings_t,
             int id )
```

This function initialize the EEPROM memory of the board by loading the default factory parameters. After the initialization a flag is set.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.

Example

```
comm_settings comm_settings_t;
int     device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

commInitMem(&comm_settings_t, device_id)

closeRS485(&comm_settings_t);
```

5.6.2.21 commPing()

```
int commPing (
              comm_settings * comm_settings_t,
             int id )
```

This function is used to ping the board.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
buffer	Buffer that stores a string with information about the device. BUFFER SIZE MUST BE AT LEAST 500.

Returns

Returns 0 if ping was ok, -1 otherwise.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
if ( commPing(&comm_settings_t, device_id) )
    puts("Device exists.");
else
    puts("Device does not exist.");

closeRS485(&comm_settings_t);
```

5.6.2.22 commRestoreParams()

```
int commRestoreParams (
              comm_settings * comm_settings_t,
             int id )
```

This function restores the factory default parameters.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.

Example

```
comm_settings comm_settings_t;
int     device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

commRestoreParams(&comm_settings_t, device_id)

closeRS485(&comm_settings_t);
```

5.6.2.23 commSetBaudRate()

```
void commSetBaudRate (
              comm_settings * comm_settings_t,
             int id,
             short int baudrate )
```

This function sets the baudrate of communication.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
baudrate	BaudRate requested 0 = 2M baudrate, 1 = 460.8k baudrate

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
short int       baudrate = 0;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
commSetBaudRate(&comm_settings_t, global_args.device_id, baudrate);
closeRS485(&comm_settings_t);
```

5.6.2.24 commSetCuffInputs()

```
void commSetCuffInputs (
              comm_settings * comm_settings_t,
             int id,
             int flag )
```

This function send reference inputs to a board connected to the serial port. Is used only when the device is a Cuff.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
flag	A flag that indicates used to activate the cuff driving functionality of the board.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
short int       cuff_inputs[2];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

int flag = 1;
commSetCuffInputs(&comm_settings_t, device_id, flag);
closeRS485(&comm_settings_t);
```

5.6.2.25 commSetInputs()

```
void commSetInputs (
              comm_settings * comm_settings_t,
             int id,
             short int inputs[] )
```

This function send reference inputs to a board connected to the serial port.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
inputs	Input references.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
short int       inputs[2];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

inputs[0]   = 1000;
inputs[1]   = -1000;
commSetInputs(&comm_settings_t, device_id, inputs);
closeRS485(&comm_settings_t);
```

5.6.2.26 commSetPosStiff()

```
void commSetPosStiff (
              comm_settings * comm_settings_t,
             int id,
             short int inputs[] )
```

This function send reference inputs to a board connected to the serial port. The reference is in shaft position and stiffness preset.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
inputs	Input references.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
short int       inputs[2];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

inputs[0]   = 100;          //Degrees
inputs[1]   = 30;           //stiffness preset
commSetPosStiff(&comm_settings_t, device_id, inputs);
closeRS485(&comm_settings_t);
```

5.6.2.27 commSetWatchDog()

```
void commSetWatchDog (
              comm_settings * comm_settings_t,
```

```
            int id,
            short int wdt )
```

This function sets watchdog timer of a board.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
wdt	Watchdog timer in [csec], max value: 500 [cs] / min value: 0 (disable) [cs]

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
short int       wdt = 60;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128);
commSetWatchDog(&comm_settings_t, global_args.device_id, wdt);
closeRS485(&comm_settings_t);
```

5.6.2.28 commSetZeros()

```
int commSetZeros (
             comm_settings * comm_settings_t,
            int id,
            void * values,
            unsigned short num_of_values )
```

This function sets the encoders's zero positon value that remains stored in the board memory.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
value	An array with the encoder readings values.
num_of_values	The size of the values array, equal to the sensor number.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
short int       measurements[3];


openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
commGetMeasurements(comm_settings_t, device_id, measurements)
for(i = 0; i<3; i++)
    measurements[i] = -measurements[i];
commSetZeros(&comm_settings_t, global_args.device_id, measurements, 3);
closeRS485(&comm_settings_t);
```

5.6.2.29 commStoreDefaultParams()

```
int commStoreDefaultParams (
              comm_settings * comm_settings_t,
             int id )
```

This function stores the factory default parameters.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.

Example

```
comm_settings comm_settings_t;
int     device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
commStoreDefaultParams(&comm_settings_t, device_id)
closeRS485(&comm_settings_t);
```

5.6.2.30 commStoreParams()

```
int commStoreParams (
              comm_settings * comm_settings_t,
             int id )
```

This function stores all parameters that were set in the board memory.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.

Example

```
comm_settings comm_settings_t;
int     device_id = 65;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");

commStoreParams(&comm_settings_t, device_id)

closeRS485(&comm_settings_t);
```

5.6.2.31 openRS485()

```
void openRS485 (
              comm_settings * comm_settings_t,
```

```
            const char * port_s,
            int BAUD_RATE = B2000000 )
```

This function is used to open a serial port for using with the board.

Parameters

comm_settings *(p. 7)*	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
port_s	The string to the serial port path.
BAUD_RATE	The default baud rate value of the serial port

Returns

Returns the file descriptor associated to the serial port.

Example

```
comm_settings   comm_settings_t;

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
if(comm_settings_t.file_handle == INVALID_HANDLE_VALUE)
{
// ERROR
}
```

5.6.2.32 RS485GetInfo()

```
void RS485GetInfo (
             comm_settings * comm_settings_t,
            char * buffer )
```

This function is used to ping the serial port for a board and to get information about the device. ONLY USE WHEN ONE DEVICE IS CONNECTED ONLY.

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
buffer	Buffer that stores a string with information about the device. BUFFER SIZE MUST BE AT LEAST 500.

Example

```
comm_settings    comm_settings_t;
char             auxstring[500];

openRS485(&comm_settings_t,"/dev/tty.usbserial-128");
RS485GetInfo(&comm_settings_t, auxstring);
puts(auxstring);
closeRS485(&comm_settings_t);
```

5.6.2.33 RS485ListDevices()

```
int RS485ListDevices (
              comm_settings * comm_settings_t,
             char list_of_ids[255] )
```

This function is used to list the number of devices connected to the serial port and get their relative IDs

Parameters

comm_↩ settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
list_of_ids[255]	Buffer that stores a list of IDs to ping, in order to see which of those IDs is connected. Is then filled with the IDs connected to the serial port.

Returns

Returns the number of devices connected

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
int             device_num;
char            list_of_ids[255];

openRS485(&comm_settings_t, device_id);
device_num = RS485ListDevices(&comm_settings_t, &list_of_ids);
closeRS485(&comm_settings_t);
printf("Number of devices connected: %d", i);
```

5.6.2.34 RS485listPorts()

```
int RS485listPorts (
             char list_of_ports[60][255] )
```

This function is used to return a list of available serial ports. A maximum of 10 ports are found.

Parameters

list_of_ports	An array of strings with the serial ports paths.

Returns

Returns the number of serial ports found.

Example

```
int     i, num_ports;
char    list_of_ports[10][255];

num_ports = RS485listPorts(ports);

for(i = 0; i < num_ports; ++i)
```

```
{
    puts(ports[i]);
}
```

5.6.2.35 RS485read()

```
int RS485read (
              comm_settings * comm_settings_t,
             int id,
             char * package )
```

This function is used to read a package from the device.

Parameters

comm_settings_t	A ***comm_settings*** *(p. 7)* structure containing info about the communication settings.
id	The device's id number.
package	Package will be stored here.

Returns

Returns package length if communication was ok, -1 otherwise.

Example

```
comm_settings   comm_settings_t;
int             device_id = 65;
char            data_read[1000];

openRS485(&comm_settings_t, "/dev/tty.usbserial-128");
commPing(&comm_settings_t, device_id);
RS485read(&comm_settings_t, device_id, data_read);
closeRS485(&comm_settings_t);

printf(data_read);
```

5.6.2.36 timevaldiff()

```
long timevaldiff (
             struct timeval * starttime,
             struct timeval * finishtime )
```

This functions returns a difference between two timeval structures in order to obtain time elapsed between the two timeval;

Parameters

starttime	The timeval structure containing the start time
finishtime	The timeval structure containing the finish time

Returns

Returns the elapsed time between the two timeval structures.

Example

```
struct timeval start, finish;
gettimeofday(&start, NULL);
// other instructions
gettimeofday(&now, NULL);
long diff = timevaldiff(&start, &now);

printf(Time elapsed: %ld, diff);
```

Index