Resistance calculation library

1

Generated by Doxygen 1.8.11

Contents

1 Readme 1

2 File Index 3

2.1 File List . . . 3

3 File Documentation 5

3.1 libresistance.c File Reference . . . 5

3.1.1 Function Documentation . . . 5

3.1.1.1 calc_resistance(int count, char conn, float *array) . . . 5

3.1.1.2 hasInvalidArguments(int count, char conn, float *array) . . . 6

3.2 libresistance.h File Reference . . . 7

3.2.1 Function Documentation . . . 7

3.2.1.1 calc_resistance(int count, char conn, float *array) . . . 7

3.3 test_main.c File Reference . . . 8

3.3.1 Enumeration Type Documentation . . . 9

3.3.1.1 boolean . . . 9

3.3.2 Function Documentation . . . 9

3.3.2.1 assertIsNotTheSame(char *testName, float expected, float given) . . . 9

3.3.2.2 assertIsTheSame(char *testName, float expected, float given) . . . 10

3.3.2.3 main() . . . 11

3.3.2.4 printFailedTestText(char *testName, char *text,...) . . . 11

3.3.2.5 printSuccessTestText(char *testName, char *text,...) . . . 12

3.3.2.6 printTestText(char *testName, char *text, char *colour, va_list args) . . . 13

Index 15

Chapter 1

Readme

Chapter 2

File Index

2.1 File List

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

libresistance.c 5
libresistance.h 7
test_main.c 8

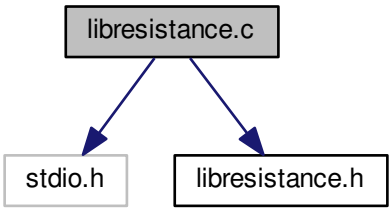
Chapter 3

File Documentation

3.1 libresistance.c File Reference

```
#include "stdio.h"
#include "libresistance.h"
```

Include dependency graph for libresistance.c:



Functions

- unsigned hasInvalidArguments (int count, char conn, float ∗array)

  Checks if any of the arguments are incorrect.
- float calc_resistance (int count, char conn, float ∗array)

  Calculate the resistance for either parallell or serial connections.

3.1.1 Function Documentation

3.1.1.1 float calc_resistance (int *count*, char *conn*, float ∗ *array*)

Calculate the resistance for either parallell or serial connections.

Calculates the resistance values for either a series of components that are connected in parallell or serial.

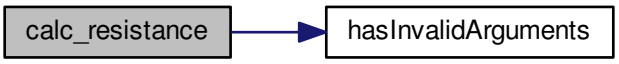
Parameters

count	The number of components
conn	The type of connection, can be either P(Parallell) or S(Serial)
∗*array*	The resistance values of the components

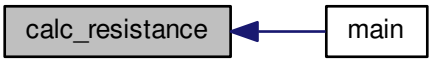
Returns

The summarized resistance value. If any of the input arguments are incorrect, -1 will be returned.

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.1.2 unsigned hasInvalidArguments (int *count*, char *conn*, float ∗ *array*)

Checks if any of the arguments are incorrect.

Checks if any of the arguments are incorrect. It checks: conn is P or S count is larger than 0 array is not an empty array

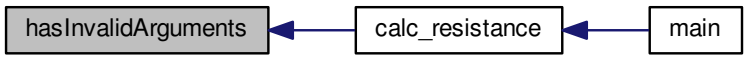
Parameters

count	The number of components
conn	The type of connection, can be either P(Parallell) or S(Serial)
∗*array*	The resistance values of the components

Returns

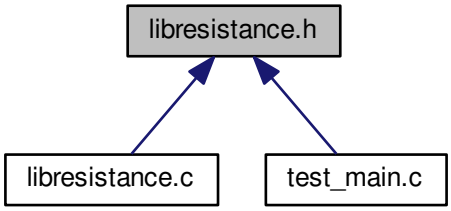
Returns -1 if any of the arguments is incorrect, otherwise 0.

Here is the caller graph for this function:



3.2 libresistance.h File Reference

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



Functions

- float calc_resistance (int count, char conn, float ∗array)

  Calculate the resistance for either parallell or serial connections.

3.2.1 Function Documentation

3.2.1.1 float calc_resistance (int *count*, char *conn*, float ∗ *array*)

Calculate the resistance for either parallell or serial connections.

Calculates the resistance values for either a series of components that are connected in parallell or serial.

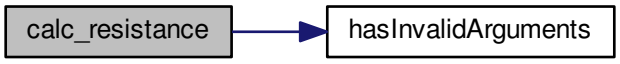
Parameters

count	The number of components
conn	The type of connection, can be either P(Parallell) or S(Serial)
∗*array*	The resistance values of the components

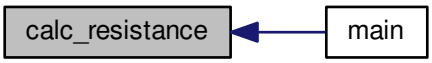
Returns

The summarized resistance value. If any of the input arguments are incorrect, -1 will be returned.

Here is the call graph for this function:



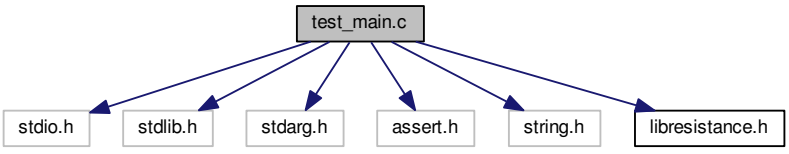
Here is the caller graph for this function:



3.3 test_main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <assert.h>
#include <string.h>
#include "libresistance.h"
```

Include dependency graph for test_main.c:



Macros

- #define ANSI_COLOR_RED "\x1b[31m"

  Defines the red colour.
- #define ANSI_COLOR_GREEN "\x1b[32m"

  Defines the green colour.

Enumerations

- enum boolean { FALSE = 0, TRUE }

  To avoid having to use 1 and 0, there is an enum definition that mimics true and false.

Functions

- void printTestText (char *testName, char *text, char *colour, va_list args)

  Prints a text with a given colour and a list of arguments.
- void printFailedTestText (char *testName, char *text,...)

  Prints failing test messages.
- void printSuccessTestText (char *testName, char *text,...)

  Prints successful test messages.
- unsigned assertIsTheSame (char *testName, float expected, float given)

  Test method that is used to check if two floats are the same.
- unsigned assertIsNotTheSame (char *testName, float expected, float given)

  Test method that is used to check if two floats are not the same.
- int main ()

  Runs all the tests.

3.3.1 Enumeration Type Documentation

3.3.1.1 enum boolean

To avoid having to use 1 and 0, there is an enum definition that mimics true and false.

Enumerator

FALSE The same as 0.

TRUE The same as 1.

3.3.2 Function Documentation

3.3.2.1 unsigned assertIsNotTheSame (char * *testName*, float *expected*, float *given*)

Test method that is used to check if two floats are not the same.

Compare two float values and returns 1 if it's not the same and 0 if it is. It allso makes an assert(expected != given).

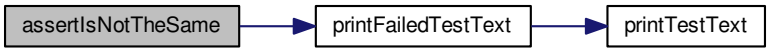
Parameters

testName	The name of the test
expected	The value to expect
given	The value returned by the test

Returns

Returns 1(TRUE) if the test is ok and 0(FALSE) if it's not

Here is the call graph for this function:



Here is the caller graph for this function:

3.3.2.2 unsigned assertIsTheSame (char * *testName*, float *expected*, float *given*)

Test method that is used to check if two floats are the same.

Compare two float values and returns 1 if it's the same and 0 if it's not. It allso makes an assert(expected == given).

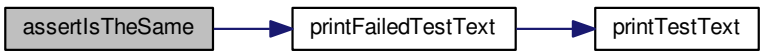
Parameters

testName	The name of the test
expected	The value to expect
given	The value returned by the test

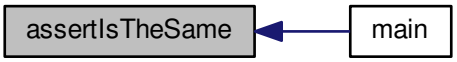
Returns

Returns 1(TRUE) if the test is ok and 0(FALSE) if it's not

Here is the call graph for this function:



Here is the caller graph for this function:



3.3.2.3 int main ( )

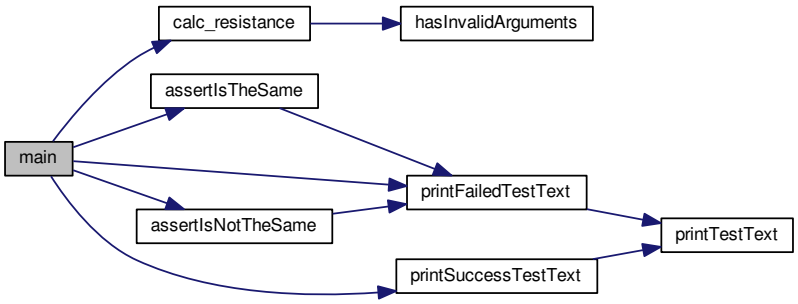
Runs all the tests.

Runs all the tests and returns 0 if it's ok and 1 if it's not

Returns

Returns 0 if it's ok and 1 if it's not ok.

Here is the call graph for this function:



3.3.2.4 void printFailedTestText (char * *testName*, char * *text*, ...)

Prints failing test messages.

Calls printTestText function with red text as pre defined and provide the parameters sent in

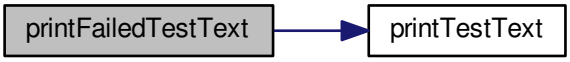
Parameters

testName	The name of the test
text	The text message to print out

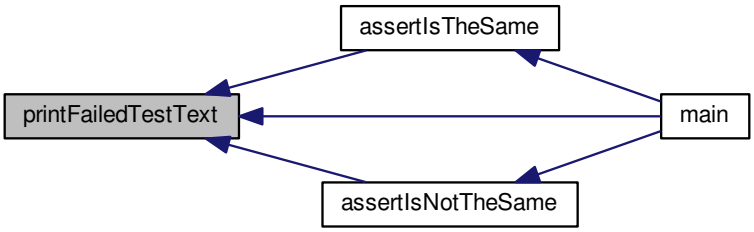
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



3.3.2.5 void printSuccessTestText (char * *testName*, char * *text*, ...)

Prints successful test messages.

Calls printTestText function with green text as pre defined and provide the parameters sent in

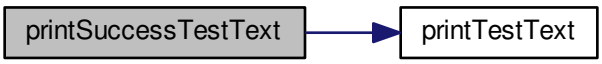
Parameters

testName	The name of the test
text	The text message to print out

Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:

3.3.2.6 void printTestText (char * *testName*, char * *text*, char * *colour*, va_list *args*)

Prints a text with a given colour and a list of arguments.

A wrapper around printf that simplifies printing out test result text in different colours.

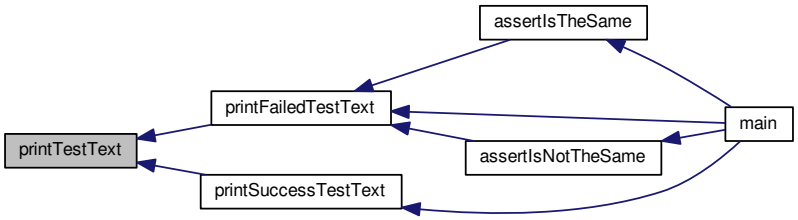
Parameters

testName	The name of the test
text	The text message to print out
colour	The colour of the text to print
args	The arguments for the text output, it's a va_list so provide it in the same way as for a printf call

Returns

void

Here is the caller graph for this function:



Index

assertIsNotTheSame
    test_main.c, 9
assertIsTheSame
    test_main.c, 10

boolean
    test_main.c, 9

calc_resistance
    libresistance.c, 5
    libresistance.h, 7

FALSE
    test_main.c, 9

hasInvalidArguments
    libresistance.c, 6

libresistance.c, 5
    calc_resistance, 5
    hasInvalidArguments, 6
libresistance.h, 7
    calc_resistance, 7

main
    test_main.c, 11

printFailedTestText
    test_main.c, 11
printSuccessTestText
    test_main.c, 12
printTestText
    test_main.c, 13

TRUE
    test_main.c, 9
test_main.c, 8
    assertIsNotTheSame, 9
    assertIsTheSame, 10
    boolean, 9
    FALSE, 9
    main, 11
    printFailedTestText, 11
    printSuccessTestText, 12
    printTestText, 13
    TRUE, 9