

# CSerialLib

0.9

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>C serial libray main page</b>	<b>1</b>
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Data Structures . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Data Structure Documentation</b>	<b>7</b>
4.1	serial_options Struct Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Field Documentation . . . . .	7
4.1.2.1	baudRate . . . . .	7
4.1.2.2	dataBits . . . . .	8
4.1.2.3	parityBit . . . . .	8
<b>5</b>	<b>File Documentation</b>	<b>9</b>
5.1	main.c File Reference . . . . .	9
5.1.1	Macro Definition Documentation . . . . .	9
5.1.1.1	MESSAGE_SIZE . . . . .	9
5.1.2	Function Documentation . . . . .	9
5.1.2.1	main(int argc, char **argv) . . . . .	9
5.2	serial.c File Reference . . . . .	9
5.2.1	Function Documentation . . . . .	10
5.2.1.1	fixed_read(int file_descriptor, size_t fixed_lenght, void *buffer) . . . . .	10

5.2.1.2	open_serial(const char name[], SerialOptions options) . . . . .	10
5.2.1.3	read_serial_packet(int file_descriptor, size_t fixed_lenght, void *buffer, uint8_t start) . . . . .	10
5.2.1.4	write_serial(int file_descriptor, size_t lenght, void *buffer) . . . . .	11
5.3	serial.h File Reference . . . . .	11
5.3.1	Macro Definition Documentation . . . . .	12
5.3.1.1	ERROR . . . . .	12
5.3.1.2	EVEN_PARITY . . . . .	12
5.3.1.3	NO_PACKET . . . . .	12
5.3.1.4	NO_PARITY . . . . .	12
5.3.1.5	ODD_PARITY . . . . .	12
5.3.2	Typedef Documentation . . . . .	12
5.3.2.1	SerialOptions . . . . .	12
5.3.3	Function Documentation . . . . .	12
5.3.3.1	fixed_read(int file_descriptor, size_t fixed_lenght, void *buffer) . . . . .	12
5.3.3.2	open_serial(const char name[], SerialOptions options) . . . . .	13
5.3.3.3	read_serial_packet(int file_descriptor, size_t fixed_lenght, void *buffer, uint8_t start) . . . . .	13
5.3.3.4	write_serial(int file_descriptor, size_t lenght, void *buffer) . . . . .	14
<b>Index</b>		<b>15</b>

## **Chapter 1**

# **C serial libray main page**

Contact `francesco.antoniazzil1991@gmail.com`



## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<b>serial_options</b>	7
-----------------------	---





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<b>main.c</b>	.....	9
<b>serial.c</b>	.....	9
<b>serial.h</b>	.....	11



## Chapter 4

# Data Structure Documentation

### 4.1 serial\_options Struct Reference

```
#include <serial.h>
```

#### Data Fields

- int **baudRate**
- int **dataBits**
- int **parityBit**

#### 4.1.1 Detailed Description

SerialOptions: a simple struct for basic setups of serial port

#### 4.1.2 Field Documentation

##### 4.1.2.1 int serial\_options::baudRate

The baud rate can be

B50, 50 baud  
B75, 75 baud  
B110, 110 baud  
B134, 134.5 baud  
B150, 150 baud  
B200, 200 baud  
B300, 300 baud  
B600, 600 baud  
B1200, 1200 baud  
B1800, 1800 baud  
B2400, 2400 baud  
B4800, 4800 baud  
B9600, 9600 baud  
B19200, 19200 baud  
B38400, 38400 baud  
B57600, 57,600 baud  
B76800, 76,800 baud  
B115200, 115,200 baud

#### 4.1.2.2 `int serial_options::dataBits`

Data bits can be  
CS5, 5 data bits  
CS6, 6 data bits  
CS7, 7 data bits  
CS8, 8 data bits

#### 4.1.2.3 `int serial_options::parityBit`

parity bit is one of the following  
NO\_PARITY  
EVEN\_PARITY  
ODD\_PARITY

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

- **serial.h**

## Chapter 5

# File Documentation

### 5.1 main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "serial.h"
```

#### Macros

- `#define MESSAGE_SIZE 15`

#### Functions

- `int main (int argc, char **argv)`

#### 5.1.1 Macro Definition Documentation

5.1.1.1 `#define MESSAGE_SIZE 15`

#### 5.1.2 Function Documentation

5.1.2.1 `int main ( int argc, char ** argv )`

### 5.2 serial.c File Reference

```
#include "serial.h"
```

## Functions

- int **open\_serial** (const char name[], **SerialOptions** options)
- int **fixed\_read** (int file\_descriptor, size\_t fixed\_lenght, void \*buffer)
- int **read\_serial\_packet** (int file\_descriptor, size\_t fixed\_lenght, void \*buffer, uint8\_t start)
- int **write\_serial** (int file\_descriptor, size\_t lenght, void \*buffer)

### 5.2.1 Function Documentation

#### 5.2.1.1 int fixed\_read ( int file\_descriptor, size\_t fixed\_lenght, void \* buffer )

fixed\_read reads a certain number of bytes from the serial port

##### Parameters

<i>file_descriptor</i>	is the identifier for the serial port
<i>fixed_lenght</i>	is the exact number of bytes to be read
<i>buffer</i>	is the place to store the bytes read

##### Returns

EXIT\_FAILURE or EXIT\_SUCCESS

#### 5.2.1.2 int open\_serial ( const char name[], SerialOptions options )

**serial.c** (p. 9)

##### Author

Francesco Antoniazzi

##### Version

0.9

##### Date

20 oct 2016

#### 5.2.1.3 int read\_serial\_packet ( int file\_descriptor, size\_t fixed\_lenght, void \* buffer, uint8\_t start )

read\_serial\_packet reads a packet from serial port. The packet is identified with a starting byte. So, while reading the first time, we ignore all is coming before this packet and then we start read packet-by-packet. The packet has a fixed lenght.

## Parameters

<i>file_descriptor</i>	is the identifier for the serial port
<i>fixed_lenght</i>	is the exact number of bytes to be read
<i>buffer</i>	is the place to store the bytes read
<i>start</i>	is the starting byte of the packet

## Returns

EXIT\_FAILURE, EXIT\_SUCCESS or NO\_PACKET

5.2.1.4 `int write_serial ( int file_descriptor, size_t lenght, void * buffer )`

`write_serial` writes to a serial port previously opened.

## Parameters

<i>file_descriptor</i>	is the identifier for the serial port
<i>lenght</i>	is the lenght of the item to be sent out
<i>buffer</i>	is a pointer to the item to be sent out

## Returns

EXIT\_FAILURE or EXIT\_SUCCESS

## 5.3 serial.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
#include <errno.h>
#include <inttypes.h>
```

## Data Structures

- struct **serial\_options**

## Macros

- #define **ERROR** -1
- #define **NO\_PACKET** -2
- #define **NO\_PARITY** 0
- #define **EVEN\_PARITY** 1
- #define **ODD\_PARITY** 2

## Typedefs

- typedef struct **serial\_options** SerialOptions

## Functions

- int **open\_serial** (const char name[], **SerialOptions** options)
- int **fixed\_read** (int file\_descriptor, size\_t fixed\_lenght, void \*buffer)
- int **read\_serial\_packet** (int file\_descriptor, size\_t fixed\_lenght, void \*buffer, uint8\_t start)
- int **write\_serial** (int file\_descriptor, size\_t lenght, void \*buffer)

### 5.3.1 Macro Definition Documentation

#### 5.3.1.1 #define ERROR -1

**serial.h** (p. 11)

#### Author

Francesco Antoniazzi

#### Version

0.9

#### Date

20 oct 2016

#### 5.3.1.2 #define EVEN\_PARITY 1

#### 5.3.1.3 #define NO\_PACKET -2

#### 5.3.1.4 #define NO\_PARITY 0

#### 5.3.1.5 #define ODD\_PARITY 2

### 5.3.2 Typedef Documentation

#### 5.3.2.1 typedef struct serial\_options SerialOptions

SerialOptions: a simple struct for basic setups of serial port

### 5.3.3 Function Documentation

#### 5.3.3.1 int fixed\_read ( int *file\_descriptor*, size\_t *fixed\_lenght*, void \* *buffer* )

fixed\_read reads a certain number of bytes from the serial port



## Parameters

<i>file_descriptor</i>	is the identifier for the serial port
<i>fixed_lenght</i>	is the exact number of bytes to be read
<i>buffer</i>	is the place to store the bytes read

## Returns

EXIT\_FAILURE or EXIT\_SUCCESS

5.3.3.2 int open\_serial ( const char *name*[], SerialOptions *options* )

open\_serial opens the serial port

## Parameters

<i>name</i> []	might be something like "/dev/ttyACM0"
<i>options</i>	is a setup for the serial port

## Returns

EXIT\_FAILURE or the serial file descriptor

**serial.c** (p. 9)

## Author

Francesco Antoniazzi

## Version

0.9

## Date

20 oct 2016

5.3.3.3 int read\_serial\_packet ( int *file\_descriptor*, size\_t *fixed\_lenght*, void \* *buffer*, uint8\_t *start* )

read\_serial\_packet reads a packet from serial port. The packet is identified with a starting byte. So, while reading the first time, we ignore all is coming before this packet and then we start read packet-by-packet. The packet has a fixed lenght.

## Parameters

<i>file_descriptor</i>	is the identifier for the serial port
<i>fixed_lenght</i>	is the exact number of bytes to be read
<i>buffer</i>	is the place to store the bytes read
<i>start</i>	is the starting byte of the packet

**Returns**

EXIT\_FAILURE, EXIT\_SUCCESS or NO\_PACKET

**5.3.3.4 int write\_serial ( int *file\_descriptor*, size\_t *length*, void \* *buffer* )**

write\_serial writes to a serial port previously opened.

**Parameters**

<i>file_descriptor</i>	is the identifier for the serial port
<i>length</i>	is the length of the item to be sent out
<i>buffer</i>	is a pointer to the item to be sent out

**Returns**

EXIT\_FAILURE or EXIT\_SUCCESS

# Index

baudRate  
    serial\_options, 7

dataBits  
    serial\_options, 7

ERROR  
    serial.h, 12

EVEN\_PARITY  
    serial.h, 12

fixed\_read  
    serial.c, 10  
    serial.h, 12

MESSAGE\_SIZE  
    main.c, 9

main  
    main.c, 9

main.c, 9  
    MESSAGE\_SIZE, 9  
    main, 9

NO\_PACKET  
    serial.h, 12

NO\_PARITY  
    serial.h, 12

ODD\_PARITY  
    serial.h, 12

open\_serial  
    serial.c, 10  
    serial.h, 13

parityBit  
    serial\_options, 8

read\_serial\_packet  
    serial.c, 10  
    serial.h, 13

serial.c, 9  
    fixed\_read, 10  
    open\_serial, 10  
    read\_serial\_packet, 10  
    write\_serial, 11

serial.h, 11  
    ERROR, 12  
    EVEN\_PARITY, 12  
    fixed\_read, 12  
    NO\_PACKET, 12

NO\_PARITY, 12  
ODD\_PARITY, 12  
open\_serial, 13  
read\_serial\_packet, 13  
SerialOptions, 12  
    write\_serial, 14  
serial\_options, 7  
    baudRate, 7  
    dataBits, 7  
    parityBit, 8  
SerialOptions  
    serial.h, 12

write\_serial  
    serial.c, 11  
    serial.h, 14