CSerialLib

0.9

Generated by Doxygen 1.8.11

Contents

Chapter 1

C serial libray main page

Contact francesco.antoniazzi1991@gmail.com

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 files with brief descriptions:

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

file_descriptor	is the identifier for the serial port
fixed_lenght	is the exact number of bytes to be read
buffer	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

file_descriptor	is the identifier for the serial port
fixed_lenght	is the exact number of bytes to be read
buffer	is the place to store the bytes read
start	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

file_descriptor	is the identifier for the serial port
lenght	is the lenght of the item to be sent out
buffer	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

file_descriptor	is the identifier for the serial port
fixed_lenght	is the exact number of bytes to be read
buffer	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

name[ ]	might be something like "/dev/ttyACM0"
options	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

file_descriptor	is the identifier for the serial port
fixed_lenght	is the exact number of bytes to be read
buffer	is the place to store the bytes read
start	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 *lenght,* void * *buffer*)

write_serial writes to a serial port previously opened.

Parameters

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

Returns

EXIT_FAILURE or EXIT_SUCCESS

Index