File Transmiter

Generated by Doxygen 1.8.16

1 Module Index 1

1.1 Modules . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 Module Documentation 5

3.1 API . . . 5

3.1.1 Detailed Description . . . 5

3.1.2 Function Documentation . . . 5

3.1.2.1 llclose() . . . 5

3.1.2.2 llopen() . . . 6

3.1.2.3 llread() . . . 6

3.1.2.4 llwrite() . . . 7

3.2 application . . . 8

3.2.1 Detailed Description . . . 8

3.2.2 Function Documentation . . . 8

3.2.2.1 build_control_package() . . . 8

3.2.2.2 build_data_package() . . . 9

3.2.2.3 process_control_package() . . . 9

3.2.2.4 process_data_package() . . . 10

3.3 protocol . . . 11

3.3.1 Detailed Description . . . 12

3.3.2 Function Documentation . . . 12

3.3.2.1 add_data_to_frame() . . . 12

3.3.2.2 addErrors() . . . 13

3.3.2.3 answer_confirmation() . . . 13

3.3.2.4 build_header() . . . 13

3.3.2.5 change_sequence_number() . . . 14

3.3.2.6 destuffing_message() . . . 14

3.3.2.7 get_sequence_answer() . . . 14

3.3.2.8 get_sequence_number() . . . 15

3.3.2.9 receive_answer() . . . 15

3.3.2.10 receive_message() . . . 15

3.3.2.11 receive_message_data() . . . 16

3.3.2.12 send_answer() . . . 16

3.3.2.13 send_message() . . . 17

3.3.2.14 send_message_data() . . . 17

3.3.2.15 send_timed_data_get_answer() . . . 17

3.3.2.16 send_timed_message_get_answer() . . . 19

3.3.2.17 serial_port_close() . . . 19

3.3.2.18 serial_port_connect() . . . 20

3.3.2.19 serial_port_end_connection() . . . 20

3.3.2.20 serial_port_start() . . . 21
3.3.2.21 serial_port_wait_connection() . . . 21
3.3.2.22 serial_port_wait_end_connection() . . . 21
3.3.2.23 set_sequence_number_rejected() . . . 22
3.3.2.24 stuffing_message() . . . 22
3.3.2.25 validate_data() . . . 23
3.3.2.26 validate_header() . . . 23
3.3.2.27 wait_for_message() . . . 23
3.4 protocol_macros . . . 25
3.4.1 Detailed Description . . . 25
3.5 state_machine . . . 26
3.5.1 Detailed Description . . . 26
3.5.2 Function Documentation . . . 26
3.5.2.1 process_state() . . . 26

4 Class Documentation **27**
4.1 application_layer Struct Reference . . . 27
4.2 link_layer Struct Reference . . . 27

Index **29**

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

API 5
application 8
protocol 11
protocol_macros 25
state_machine 26

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

application_layer 27
link_layer 27

Chapter 3

Module Documentation

3.1 API

Funcitons that interact with the protocol.

Classes

- struct link_layer

Macros

- #define **_TRANSMITTER** 0
- #define **_RECEIVER** 1

Functions

- int llopen (int port, int device)

  Opens the serial port and tries to establish a connection.
- int llwrite (int fd, unsigned char ∗buffer, int length)

  Writes the message to fd.
- int llread (int fd, unsigned char ∗buffer)

  Reads the message from fd.
- int llclose (int port_fd)

  Ends a connection and closes the serial port.

3.1.1 Detailed Description

Funcitons that interact with the protocol.

3.1.2 Function Documentation

3.1.2.1 llclose()

```
int llclose (
            int port_fd )
```

Ends a connection and closes the serial port.

Parameters

port_fd	- File descriptor of the port

Returns

int - positive if OK, negative if error

3.1.2.2 llopen()

```
int llopen (
            int port,
            int device )
```

Opens the serial port and tries to establish a connection.

Parameters

port	- Number of the port to open (COM1 OR COM2)
buffer	- output: the data received through

Returns

int - file descriptor of connection, negative value on error

3.1.2.3 llread()

```
int llread (
            int fd,
            unsigned char * buffer )
```

Reads the message from fd.

Parameters

port	- file descriptor from where the data will be read
buffer	- data that was received

Returns

int - number of chars read (size of buffer)

3.1.2.4 llwrite()

```
int llwrite (
             int fd,
             unsigned char * buffer,
             int length )
```

Writes the message to fd.

Parameters

fd	- file descriptor from where the data will be sent
buffer	- data to be written
length	- size of buffer

Returns

int - number of bytes written (size of buffer)

3.2 application

Application layer funtions.

Classes

- struct application_layer

Functions

- int build_control_package (unsigned char control, struct application_layer *info, unsigned char package[ ])

  Creates a control package with the specified fields.
- int build_data_package (int sequence_num, int data_length, unsigned char data[ ], unsigned char package[ ])

  Creates a data package with the specified fields.
- int process_data_package (unsigned char data_package[ ], unsigned char extracted_data[ ], int *sequence_num)

  Processes a received data package.
- int process_control_package (unsigned char package[ ], struct application_layer *info)

  Processes a received control package.

3.2.1 Detailed Description

Application layer funtions.

3.2.2 Function Documentation

3.2.2.1 build_control_package()

```
int build_control_package (
            unsigned char control,
            struct application_layer * info,
            unsigned char package[] )
```

Creates a control package with the specified fields.

Parameters

control	- The control byte of the package
info	- Information about the file being trasfered
package	- Array where the package will be put

Returns

int - The size of the array

3.2.2.2 build_data_package()

```
int build_data_package (
            int sequence_num,
            int data_length,
            unsigned char data[],
            unsigned char package[] )
```

Creates a data package with the specified fields.

Parameters

sequence_num	- Sequence number of the package
data_length	- Number of the bytes of the data in the package
data	- Data to include in the package
package	- Array where to put the package

Returns

int - Size of the array

3.2.2.3 process_control_package()

```
int process_control_package (
            unsigned char package[],
            struct application_layer * info )
```

Processes a received control package.

Parameters

package	- The array where the package is stored
info	- Information about the file being trasfered

Returns

int - The command or -1 on error

3.2.2.4 process_data_package()

```
int process_data_package (
            unsigned char data_package[],
            unsigned char extracted_data[],
            int * sequence_num )
```

Processes a received data package.

Parameters

data_package	- The array where the package is stored
extracted_data	- The array where the data will be stored
sequence_num	- Pointer to an integer where the sequence number of the package will be stored

Returns

int - The length of the data processed

3.3 protocol

Functions that interact directly with the serial port.

Functions

- int change_sequence_number ()

  defines the current iteration sequence number, setting it to be the last one sent and returning the new sequence number
- int get_sequence_number (unsigned char ∗msg)

  Returns the sequence number of the message.
- int get_sequence_answer (unsigned char ∗msg)

  Returns the sequence number of the answer.
- int set_sequence_number_rejected (unsigned char ∗msg)

  TRANSMITTER: sets last sequence number dependent on the received answer.
- int receive_message (int serial_port, unsigned char ∗msg_ptr)

  Receives information through the serial port.
- int send_message (int serial_port, unsigned char ∗sent_msg, unsigned char addr, unsigned char c_set)

  Sends a command through the serial port.
- int wait_for_message (int serial_port, unsigned char rec_addr, unsigned char rec_c_set)

  Waits to receive a command.
- int send_timed_message_get_answer (int serial_port, unsigned char addr, unsigned char c_set, unsigned char ans_addr, unsigned char ans_c_set, int tries, int timeout_time)

  Sends a message and waits for an answer, if the no answer received or the answer is wrong sends the message again.
- int serial_port_connect (int serial_port, int tries, int timeout)

  Sends the SET command through the serial port and waits for an UA.
- int serial_port_wait_connection (int serial_port)

  Waits for a SET command and sends a UA command.
- int serial_port_start (char ∗serial_port_name, speed_t baudrate, struct termios ∗newtio, struct termios ∗oldtio)

  Opens a serial port and sets its configuration.
- int serial_port_close (int serial_port_fd, struct termios ∗oldtio)

  Closes the serial port setting its configuration to oldtio.
- int serial_port_end_connection (int serial_port, int tries, int timeout)

  Ends a serial port connection.
- int serial_port_wait_end_connection (int serial_port, int tries, int timeout)

  Waits the command to end a connection.
- int build_header (unsigned char ∗msg)

  receives the empty message and startes building the frame
- int add_data_to_frame (unsigned char ∗msg, unsigned char ∗data, int data_length)

  Adds input to the output msg.
- int stuffing_message (unsigned char ∗output, unsigned char ∗input, int input_length)

  stuffes the data bytes that are equal to special bytes, except the initial and final FLAG itself
- int send_message_data (int serial_port, unsigned char ∗data, int data_length)

  Sends data message through the serial port.
- int answer_confirmation (unsigned char ∗msg)

  Verifies if answer is positive and not duplicated.
- int receive_answer (int serial_port)

  Receives answer to sent message and verifies if answer is accordingly to sent.
- int send_timed_data_get_answer (int serial_port, unsigned char ∗data, int data_length)

Sends a data message (I) and waits for an answer, if nop answer received or the answer is wrong sends the message again.

- int destuffing_message (unsigned char *output, unsigned char *input, int input_length)

  applies destuffing operation to the message received
- int validate_header (unsigned char *msg)

  Verifies the header info validity.
- int validate_data (unsigned char *msg, int msg_length)

  Verifies the data info validity.
- int send_answer (int serial_port, unsigned char *msg, int msg_length)

  Answers sender, accordingly to the message received.
- int receive_message_data (int serial_port, unsigned char *data_msg)

  Receives one data frame through serial port.
- void addErrors (unsigned char *msg, int length)

  Introduces bit flips in the frame received, so the code can be tested, the approval functions must return errors when bit flips happen has as internal argument that when changed, provide a ratio of error in the frame received.

3.3.1 Detailed Description

Functions that interact directly with the serial port.

3.3.2 Function Documentation

3.3.2.1 add_data_to_frame()

```
int add_data_to_frame (
            unsigned char * msg,
            unsigned char * data,
            int data_length )
```

Adds input to the output msg.

Parameters

msg	- msg to be field with data
data	- data to be put in msg
data_length	- size of data argument

Returns

size of msg or negative if error

3.3.2.2 addErrors()

```
void addErrors (
            unsigned char * msg,
            int length )
```

Introduces bit flips in the frame received, so the code can be tested, the approval functions must return errors when bit flips happen has as internal argument that when changed, provide a ratio of error in the frame received.

Parameters

msg	- message to be altered
length	- size of the msg

Returns

void

3.3.2.3 answer_confirmation()

```
int answer_confirmation (
            unsigned char * msg )
```

Verifies if answer is positive and not duplicated.

Parameters

msg	- answer to be analised

Returns

int - 1 if OK or 0 if NOT OK

3.3.2.4 build_header()

```
int build_header (
            unsigned char * msg )
```

receives the empty message and startes building the frame

TRANSMITTER END

Parameters

msg	- empty message to be constructed

Returns

int - 0 if OK or negative if errors occured

3.3.2.5 change_sequence_number()

```
int change_sequence_number ( )
```

defines the current iteration sequence number, setting it to be the last one sent and returning the new sequence number

Returns

int - the current sequence number

3.3.2.6 destuffing_message()

```
int destuffing_message (
            unsigned char * output,
            unsigned char * input,
            int input_length )
```

applies destuffing operation to the message received

RECEIVER END

Parameters

output	- destuffed message (MUST BE INITIALIZED)
input	- stuffed message
input_length	- size of input

Returns

int - size of output and -1 if error

3.3.2.7 get_sequence_answer()

```
int get_sequence_answer (
            unsigned char * msg )
```

Returns the sequence number of the answer.

Parameters

msg	- answer to be analised

Returns

int - the respective sequence number [0 OR 1] of msg

3.3.2.8 get_sequence_number()

```
int get_sequence_number (
            unsigned char * msg )
```

Returns the sequence number of the message.

Parameters

msg	- message to be analised

Returns

int - the respective sequence number [0 OR 1] of msg

3.3.2.9 receive_answer()

```
int receive_answer (
            int serial_port )
```

Receives answer to sent message and verifies if answer is accordingly to sent.

Returns

positive if OK or negative on error (needs to repeat sending)

3.3.2.10 receive_message()

```
int receive_message (
            int serial_port,
            unsigned char * msg_ptr )
```

Receives information through the serial port.

Parameters

serial_port	- The serial file descriptor of the serial port
msg_ptr	- The array in which to store the message

Returns

int - number of bytes received or -1 if some error ocurred

3.3.2.11 receive_message_data()

```
int receive_message_data (
            int serial_port,
            unsigned char * data_msg )
```

Receives one data frame through serial port.

Parameters

serial_port	- file descriptor from where the message arrives
data_msg	- output, returns the data received

Returns

int - size of data_msg (all correct), 0 if disconnect command, -1 if message should be ignored (some processing or header error)

3.3.2.12 send_answer()

```
int send_answer (
            int serial_port,
            unsigned char * msg,
            int msg_length )
```

Answers sender, accordingly to the message received.

Parameters

serial_port	- port that the answer will be sent
msg	- message to be analised @ return int - 0 if OK or -1 if REJ or repeated (data should be ignored)

3.3.2.13 send_message()

```
int send_message (
            int serial_port,
            unsigned char * sent_msg,
            unsigned char addr,
            unsigned char c_set )
```

Sends a command through the serial port.

Parameters

serial_port	- The serial port to send the message
sent_msg	- Array where to store the message sent
aflag	- The address of the sender
c_set	- The command to send

Returns

int - size of sent message or negative on error

3.3.2.14 send_message_data()

```
int send_message_data (
            int serial_port,
            unsigned char * data,
            int data_length )
```

Sends data message through the serial port.

Parameters

serial_port	- The serial port to send the message
data	- The data to be sent
data_length	- size of data argument

Returns

int - number of data bytes sent, negative if error

3.3.2.15 send_timed_data_get_answer()

```
int send_timed_data_get_answer (
            int serial_port,
```

```
unsigned char * data,
int data_length )
```

Sends a data message (I) and waits for an answer, if nop answer received or the answer is wrong sends the message again.

Parameters

serial_port	- the file descriptor of the serial port
data	- data to be sent
data_length	- size of data

Returns

int - positive if message was send and received expected answer, -1 on timeout

3.3.2.16 send_timed_message_get_answer()

```
int send_timed_message_get_answer (
            int serial_port,
            unsigned char addr,
            unsigned char c_set,
            unsigned char ans_addr,
            unsigned char ans_c_set,
            int tries,
            int timeout_time )
```

Sends a message and waits for an answer, if the no answer received or the answer is wrong sends the message again.

Parameters

serial_port	- The file descriptor of the serial port
addr	- The address of the command to send
c_set	- The command to send
ans_addr	- The address expected to receive
ans_c_set	- The command expected to receive
tries	- Number of tries until a timeout
timeout_time	- The time between each try

Returns

int - positive if message was sent and received expected answer, -1 on timeout

3.3.2.17 serial_port_close()

```
int serial_port_close (
            int serial_port_fd,
            struct termios * oldtio )
```

Closes the serial port setting its configuration to oldtio.

Parameters

serial_port↩ _fd	- File descriptor where the serial port is open
oldtio	- Configuration to set the serial port (Usually the configuration it had before startig)

Returns

int - only 0 for now

3.3.2.18 serial_port_connect()

```
int serial_port_connect (
            int serial_port,
            int tries,
            int timeout )
```

Sends the SET command through the serial port and waits for an UA.

Parameters

serial_port	- file descritor of the serial port to connect
tries	- number of tries in case of a timeout
timeout	- number of seconds until a timeout

Returns

int - positive if OK or negative if timeout ocurred

3.3.2.19 serial_port_end_connection()

```
int serial_port_end_connection (
            int serial_port,
            int tries,
            int timeout )
```

Ends a serial port connection.

Parameters

serial_port	- File escriptor of the serial port
tries	- Number of tries until timeout
timeout	- Time between tries

Returns

int - positive if OK or negative on error

3.3.2.20 serial_port_start()

```
int serial_port_start (
            char * serial_port_name,
            speed_t baudrate,
            struct termios * newtio,
            struct termios * oldtio )
```

Opens a serial port and sets its configuration.

Parameters

serial_port_name	- String containing the file of the driver Ex: "/dev/ttyS1"
baudrate	- Baudrate to set the port
newtio	- Pointer to a termios struct to store the new termios config
oldtio	- Pointer to a termios struct to store the old termios config

Returns

int - File descriptor where the serial port is open

3.3.2.21 serial_port_wait_connection()

```
int serial_port_wait_connection (
            int serial_port )
```

Waits for a SET command and sends a UA command.

Parameters

serial_port	- The serial port to listen for a connection

Returns

int - positive on success or negative on error

3.3.2.22 serial_port_wait_end_connection()

```
int serial_port_wait_end_connection (
            int serial_port,
```

```
            int tries,
            int timeout )
```

Waits the command to end a connection.

Parameters

serial_port	- File descriptor of the serial port
tries	- Number of tries until a timeout
timeout	- Time between each try

Returns

int - positive if OK or negative on error

3.3.2.23 set_sequence_number_rejected()

```
int set_sequence_number_rejected (
            unsigned char * msg )
```

TRANSMITTER: sets last sequence number dependent on the received answer.

Parameters

msg	- message to be analised

Returns

int - new last sequence number

3.3.2.24 stuffing_message()

```
int stuffing_message (
            unsigned char * output,
            unsigned char * input,
            int input_length )
```

stuffes the data bytes that are equal to special bytes, except the initial and final FLAG itself

Parameters

output	- message with the stuffing applied
input	- message to be applied the stuffing
input_length	- size of input

Returns

int - size of output

3.3.2.25 validate_data()

```
int validate_data (
            unsigned char * msg,
            int msg_length )
```

Verifies the data info validity.

Parameters

msg	- message to be analised
msg_length	- size of msg

Returns

int - 1 if OK, 0 if not valid

3.3.2.26 validate_header()

```
int validate_header (
            unsigned char * msg )
```

Verifies the header info validity.

Parameters

msg	- message to be analised

Returns

int - 0 if OK, -1 if not valid header

3.3.2.27 wait_for_message()

```
int wait_for_message (
            int serial_port,
            unsigned char rec_addr,
            unsigned char rec_c_set )
```

Waits to receive a command.

Parameters

serial_port	- The file descriptor of the serial port
rec_addr	- The addr expecting to receive
rec_c_set	- The command expecting to receive

Returns

int - positive if receives correct message or negative on error

3.4 protocol_macros

Macros used by the protocol.

Macros

- #define **TRUE** 1
- #define **FALSE** 0
- #define **MAX_FRAME_SIZE** 4096
- #define **TIMEOUT_TRIES** 25
- #define **TIMEOUT_TIME** 2
- #define **SET** 0x03
- #define **DISC** 0x0b
- #define **UA** 0x07
- #define **RR** 0x05
- #define **REJ** 0x01
- #define **FLAG** 0x7e
- #define **SENDER** 0x03
- #define **RECEIVER** 0x01
- #define **BAUDRATE** B38400
- #define **ESCAPE** 0x7d
- #define **ESC_BYTE** 0x20
- #define **I0** 0x00
- #define **I1** 0x40
- #define **R0** 0x00
- #define **R1** 0x80

3.4.1 Detailed Description

Macros used by the protocol.

3.5 state_machine

Processing of the state machine.

Enumerations

- enum **State** {
  INIT, **FLG**, **ADDR**, **CTRL**,
  PROT, **DATA**, **FINAL**, **ERROR** }

Functions

- enum State process_state (enum State current_state, unsigned char received, unsigned char ∗msg, int ∗current_byte)

  Processes a state, returns the next state considering the current state and the received character.

3.5.1 Detailed Description

Processing of the state machine.

3.5.2 Function Documentation

3.5.2.1 process_state()

```
enum State process_state (
            enum State current_state,
            unsigned char received,
            unsigned char * msg,
            int * current_byte )
```

Processes a state, returns the next state considering the current state and the received character.

Parameters

current_state	- State enum indicating the current state
received	- The byte of the frame we want to process
msg	- array where to store the bytes of the message
current_byte	- current position to put byte in the msg

Returns

enum State - returns next state

Chapter 4

Class Documentation

4.1 application_layer Struct Reference

Public Attributes

- int **port_file_des**
- int **file_des**
- char **file_name** [MAX_FILE_N]
- int **file_size**
- int **device**
- int **port**

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

- application.h

4.2 link_layer Struct Reference

Public Attributes

- char **port** [20]
- int **file_des**
- speed_t **baud_rate**
- int **device**
- unsigned int **timeout**
- unsigned int **num_transmit**
- struct termios **oldterm**
- struct termios **newterm**

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

- api.h

Index

add_data_to_frame
    protocol, 12
addErrors
    protocol, 12
answer_confirmation
    protocol, 13
API, 5
    llclose, 5
    llopen, 6
    llread, 6
    llwrite, 6
application, 8
    build_control_package, 8
    build_data_package, 9
    process_control_package, 9
    process_data_package, 9
application_layer, 27

build_control_package
    application, 8
build_data_package
    application, 9
build_header
    protocol, 13

change_sequence_number
    protocol, 14

destuffing_message
    protocol, 14

get_sequence_answer
    protocol, 14
get_sequence_number
    protocol, 15

link_layer, 27
llclose
    API, 5
llopen
    API, 6
llread
    API, 6
llwrite
    API, 6

process_control_package
    application, 9
process_data_package
    application, 9
process_state
    state_machine, 26
protocol, 11
    add_data_to_frame, 12
    addErrors, 12
    answer_confirmation, 13
    build_header, 13
    change_sequence_number, 14
    destuffing_message, 14
    get_sequence_answer, 14
    get_sequence_number, 15
    receive_answer, 15
    receive_message, 15
    receive_message_data, 16
    send_answer, 16
    send_message, 16
    send_message_data, 17
    send_timed_data_get_answer, 17
    send_timed_message_get_answer, 19
    serial_port_close, 19
    serial_port_connect, 20
    serial_port_end_connection, 20
    serial_port_start, 21
    serial_port_wait_connection, 21
    serial_port_wait_end_connection, 21
    set_sequence_number_rejected, 22
    stuffing_message, 22
    validate_data, 23
    validate_header, 23
    wait_for_message, 23
protocol_macros, 25

receive_answer
    protocol, 15
receive_message
    protocol, 15
receive_message_data
    protocol, 16

send_answer
    protocol, 16
send_message
    protocol, 16
send_message_data
    protocol, 17
send_timed_data_get_answer
    protocol, 17
send_timed_message_get_answer
    protocol, 19
serial_port_close
    protocol, 19

serial_port_connect
protocol, 20
serial_port_end_connection
protocol, 20
serial_port_start
protocol, 21
serial_port_wait_connection
protocol, 21
serial_port_wait_end_connection
protocol, 21
set_sequence_number_rejected
protocol, 22
state_machine, 26
process_state, 26
stuffing_message
protocol, 22

validate_data
protocol, 23
validate_header
protocol, 23

wait_for_message
protocol, 23