

MaschinenRata

Generated by Doxygen 1.8.13

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	linesensors_t Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	5
3.1.2.1	left	5
3.1.2.2	mid	6
3.1.2.3	midl	6
3.1.2.4	midr	6
3.1.2.5	right	6
3.2	motor_t Struct Reference	6
3.2.1	Field Documentation	6
3.2.1.1	direction	6
3.2.1.2	speed	7
3.3	uart_buffer Struct Reference	7
3.3.1	Field Documentation	7
3.3.1.1	buffer	7
3.3.1.2	read	7
3.3.1.3	write	7

4	File Documentation	9
4.1	include/config.h File Reference	9
4.1.1	Macro Definition Documentation	9
4.1.1.1	F_CPU	9
4.1.1.2	UART_BAUDRATE	9
4.1.1.3	UART_BUFFER_SIZE	9
4.2	include/hardware/linesensors.h File Reference	10
4.2.1	Detailed Description	10
4.2.2	Typedef Documentation	11
4.2.2.1	line_t	11
4.2.3	Enumeration Type Documentation	11
4.2.3.1	line_status	11
4.2.4	Function Documentation	11
4.2.4.1	linesensors_get()	12
4.2.4.2	linesensors_init()	12
4.2.4.3	linesensors_set_threshold()	12
4.2.4.4	linesensors_update()	12
4.2.5	Variable Documentation	13
4.2.5.1	linesensors	13
4.3	include/hardware/motors.h File Reference	13
4.3.1	Detailed Description	14
4.3.2	Macro Definition Documentation	14
4.3.2.1	MAX_SPEED	14
4.3.2.2	ROTATE_SPEED	14
4.3.3	Function Documentation	14
4.3.3.1	motors_forward()	14
4.3.3.2	motors_init()	14
4.3.3.3	motors_lean_neg()	14
4.3.3.4	motors_lean_pos()	15
4.3.3.5	motors_reverse()	15

4.3.3.6	motors_rotate_neg()	15
4.3.3.7	motors_rotate_pos()	15
4.3.3.8	motors_stop()	15
4.3.3.9	motors_update()	15
4.3.4	Variable Documentation	15
4.3.4.1	motor_left	15
4.3.4.2	motor_right	16
4.4	include/hardware/uart.h File Reference	16
4.4.1	Macro Definition Documentation	16
4.4.1.1	UART_DEMULT	16
4.4.2	Function Documentation	16
4.4.2.1	uart_available()	17
4.4.2.2	uart_find()	17
4.4.2.3	uart_getc()	17
4.4.2.4	uart_gets()	17
4.4.2.5	uart_init()	17
4.4.2.6	uart_putc()	17
4.4.2.7	uart_puts()	17
4.4.2.8	uart_read()	18
4.4.2.9	uart_read_until()	18
4.4.2.10	uart_readline()	18
4.4.2.11	uart_write()	18
Index		19

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

linesensors_t	Stores the raw sensor data	5
motor_t	6
uart_buffer	7

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

include/ config.h	9
include/hardware/ linesensors.h	
Functions to control the linesensors	10
include/hardware/ motors.h	
This controls the motors of the robot	13
include/hardware/ uart.h	16

Chapter 3

Data Structure Documentation

3.1 linesensors_t Struct Reference

Stores the raw sensor data.

```
#include <linesensors.h>
```

Data Fields

- uint8_t [left](#)
- uint8_t [midl](#)
- uint8_t [mid](#)
- uint8_t [midr](#)
- uint8_t [right](#)

3.1.1 Detailed Description

Stores the raw sensor data.

This structure stores the integer value of the sensors, in integers with 8bit size.

3.1.2 Field Documentation

3.1.2.1 left

```
uint8_t linesensors_t::left
```

The left sensor output

3.1.2.2 mid

```
uint8_t linesensors_t::mid
```

The mid-mid sensor output

3.1.2.3 midl

```
uint8_t linesensors_t::midl
```

The mid-left sensor output

3.1.2.4 midr

```
uint8_t linesensors_t::midr
```

The mid-right sensor output

3.1.2.5 right

```
uint8_t linesensors_t::right
```

The right sensor output

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

- include/hardware/[linesensors.h](#)

3.2 motor_t Struct Reference

```
#include <motors.h>
```

Data Fields

- uint8_t [speed](#)
- uint8_t [direction](#)

3.2.1 Field Documentation

3.2.1.1 direction

```
uint8_t motor_t::direction
```

the direction of rotation of the motor. a zero is cw, else if ccw

3.2.1.2 speed

```
uint8_t motor_t::speed
```

the speed of the motor (can be a number between 0-255)

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

- `include/hardware/motors.h`

3.3 uart_buffer Struct Reference

```
#include <uart.h>
```

Data Fields

- `uint8_t buffer [UART_BUFFER_SIZE]`
- `uint8_t write`
- `uint8_t read`

3.3.1 Field Documentation

3.3.1.1 buffer

```
uint8_t uart_buffer::buffer [UART_BUFFER_SIZE]
```

3.3.1.2 read

```
uint8_t uart_buffer::read
```

3.3.1.3 write

```
uint8_t uart_buffer::write
```

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

- `include/hardware/uart.h`

Chapter 4

File Documentation

4.1 include/config.h File Reference

Macros

- `#define F_CPU 16000000L`
- `#define UART_BAUDRATE 9600`
- `#define UART_BUFFER_SIZE 64`

4.1.1 Macro Definition Documentation

4.1.1.1 F_CPU

```
#define F_CPU 16000000L
```

4.1.1.2 UART_BAUDRATE

```
#define UART_BAUDRATE 9600
```

4.1.1.3 UART_BUFFER_SIZE

```
#define UART_BUFFER_SIZE 64
```

4.2 include/hardware/linesensors.h File Reference

Functions to control the linesensors.

```
#include <stdint.h>
```

Data Structures

- struct [linesensors_t](#)
Stores the raw sensor data.

Typedefs

- typedef uint8_t [line_t](#)
Stores the boolean state of the line sensors.

Enumerations

- enum [line_status](#) {
 [ALL_OFF](#) = 0b00000, [ALL_ON](#) = 0b11111, [ON_TRACK](#) = 0b00100, [CENTERED](#) = 0b01110,
 [RIGHT_TILTED](#) = 0b01000, [LEFT_TILTED](#) = 0b00010, [RIGHT](#) = 0b10000, [LEFT](#) = 0b00001 }
Stores recurrent sensor status.

Functions

- void [linesensors_init](#) ()
Initializes the line sensors.
- void [linesensors_update](#) ()
Reads all the sensors and updates [linesensors](#).
- void [linesensors_set_threshold](#) ([linesensors_t](#) threshold)
Sets the threshold for each sensor.
- [line_t](#) [linesensors_get](#) ()
Get the boolean values of the sensors.

Variables

- [linesensors_t](#) [linesensors](#)
Variable that stores the current value of the sensors.

4.2.1 Detailed Description

Functions to control the linesensors.

This contains all the functions to interface with the line sensors. Usage example:

```
line_t ls = linesensors_get();  
if (ls & RIGHT) {  
    // do something, like rotating right.  
}
```


4.2.2 Typedef Documentation

4.2.2.1 line_t

```
typedef uint8_t line_t
```

Stores the boolean state of the line sensors.

Each bit of this integer represents a sensor state. The most significant is the right sensor and the least significant is the left sensor.

4.2.3 Enumeration Type Documentation

4.2.3.1 line_status

```
enum line_status
```

Stores recurrent sensor status.

A enumeration of useful constants that can be used to check for recurrent sensor states. Usually is used in conjunction with the binary and (&) to check the state, for example,

```
linesensor_get() & LEFT
```

checks if the left sensor is triggered.

Enumerator

ALL_OFF	All sensors are off.
ALL_ON	All sensors are on.
ON_TRACK	The mid sensor is on.
CENTERED	The mid sensors are on.
RIGHT_TILTED	The mid-right sensor is on.
LEFT_TILTED	The mid-left sensor is on.
RIGHT	The right sensor is on.
LEFT	The left sensor is on.

4.2.4 Function Documentation

4.2.4.1 linesensors_get()

```
line_t linesensors_get ( )
```

Get the boolean values of the sensors.

Returns

the boolean output.

4.2.4.2 linesensors_init()

```
void linesensors_init ( )
```

Initializes the line sensors.

Initializes the ADC of the m328p to read the analog values of the pins PB0-PB4. The following configurations were done:

- resolution of the ADC: 8bit;
- clock divider: CLK/256;

4.2.4.3 linesensors_set_threshold()

```
void linesensors_set_threshold (
    linesensors_t threshold )
```

Sets the threshold for each sensor.

When the function [linesensors_get](#) is called, this threshold level is used to create the boolean output of each sensor. Each sensor value is compared to the threshold, and is evaluated to true if it is bigger than the threshold. There are 5 thresholds, each one for each sensor, for better control.

Parameters

<i>threshold</i>	The threshold value, that will be copied to a static variable.
------------------	--

4.2.4.4 linesensors_update()

```
void linesensors_update ( )
```

Reads all the sensors and updates [linesensors](#).

All the line sensors are read sequentially from the left to the right and the raw value of each is stored in the global variable `linesensors`.

4.2.5 Variable Documentation

4.2.5.1 linesensors

```
linesensors_t linesensors
```

Variable that stores the current value of the sensors.

Global variable that stores the current value of the sensors. Only updates when a call to `linesensors_update` is done.

4.3 include/hardware/motors.h File Reference

This controls the motors of the robot.

```
#include <avr/io.h>
```

Data Structures

- struct `motor_t`

Macros

- `#define MAX_SPEED 120`
- `#define ROTATE_SPEED 100`

Functions

- void `motors_init ()`
- void `motors_update ()`
- void `motors_forward ()`
- void `motors_reverse ()`
- void `motors_stop ()`
- void `motors_lean_pos ()`
- void `motors_lean_neg ()`
- void `motors_rotate_pos ()`
- void `motors_rotate_neg ()`

Variables

- [motor_t motor_left](#)
- [motor_t motor_right](#)

4.3.1 Detailed Description

This controls the motors of the robot.

4.3.2 Macro Definition Documentation

4.3.2.1 MAX_SPEED

```
#define MAX_SPEED 120
```

4.3.2.2 ROTATE_SPEED

```
#define ROTATE_SPEED 100
```

4.3.3 Function Documentation

4.3.3.1 motors_forward()

```
void motors_forward ( )
```

4.3.3.2 motors_init()

```
void motors_init ( )
```

4.3.3.3 motors_lean_neg()

```
void motors_lean_neg ( )
```

4.3.3.4 motors_lean_pos()

```
void motors_lean_pos ( )
```

4.3.3.5 motors_reverse()

```
void motors_reverse ( )
```

4.3.3.6 motors_rotate_neg()

```
void motors_rotate_neg ( )
```

4.3.3.7 motors_rotate_pos()

```
void motors_rotate_pos ( )
```

4.3.3.8 motors_stop()

```
void motors_stop ( )
```

4.3.3.9 motors_update()

```
void motors_update ( )
```

4.3.4 Variable Documentation

4.3.4.1 motor_left

```
motor_t motor_left
```

4.3.4.2 motor_right

```
motor_t motor_right
```

4.4 include/hardware/uart.h File Reference

```
#include "config.h"  
#include <avr/io.h>
```

Data Structures

- struct [uart_buffer](#)

Macros

- #define [UART_DEMULT](#) ([F_CPU](#) / 16 / [UART_BAUDRATE](#) - 1)

Functions

- void [uart_init](#) ()
- uint8_t [uart_write](#) (uint8_t, uint8_t *)
- uint8_t [uart_read](#) (uint8_t, uint8_t *)
- uint8_t [uart_read_until](#) (uint8_t, uint8_t *, uint8_t)
- void [uart_putc](#) (char)
- uint8_t [uart_puts](#) (char *)
- uint8_t [uart_available](#) ()
- uint8_t [uart_find](#) (char)
- char [uart_getc](#) ()
- uint8_t [uart_gets](#) (uint8_t, char *)
- uint8_t [uart_readline](#) (uint8_t, char *)

4.4.1 Macro Definition Documentation

4.4.1.1 UART_DEMULT

```
#define UART_DEMULT (F_CPU / 16 / UART_BAUDRATE - 1)
```

4.4.2 Function Documentation

4.4.2.1 uart_available()

```
uint8_t uart_available ( )
```

4.4.2.2 uart_find()

```
uint8_t uart_find (
    char )
```

4.4.2.3 uart_getc()

```
char uart_getc ( )
```

4.4.2.4 uart_gets()

```
uint8_t uart_gets (
    uint8_t ,
    char * )
```

4.4.2.5 uart_init()

```
void uart_init ( )
```

4.4.2.6 uart_putc()

```
void uart_putc (
    char )
```

4.4.2.7 uart_puts()

```
uint8_t uart_puts (
    char * )
```

4.4.2.8 uart_read()

```
uint8_t uart_read (
    uint8_t ,
    uint8_t * )
```

4.4.2.9 uart_read_until()

```
uint8_t uart_read_until (
    uint8_t ,
    uint8_t * ,
    uint8_t )
```

4.4.2.10 uart_readline()

```
uint8_t uart_readline (
    uint8_t ,
    char * )
```

4.4.2.11 uart_write()

```
uint8_t uart_write (
    uint8_t ,
    uint8_t * )
```


Index

buffer
 uart_buffer, 7

config.h
 F_CPU, 9
 UART_BAUDRATE, 9
 UART_BUFFER_SIZE, 9

direction
 motor_t, 6

F_CPU
 config.h, 9

include/config.h, 9
include/hardware/linesensors.h, 10
include/hardware/motors.h, 13
include/hardware/uart.h, 16

left
 linesensors_t, 5

line_status
 linesensors.h, 11

line_t
 linesensors.h, 11

linesensors
 linesensors.h, 13

linesensors.h
 line_status, 11
 line_t, 11
 linesensors, 13
 linesensors_get, 11
 linesensors_init, 12
 linesensors_set_threshold, 12
 linesensors_update, 12

linesensors_get
 linesensors.h, 11

linesensors_init
 linesensors.h, 12

linesensors_set_threshold
 linesensors.h, 12

linesensors_t, 5
 left, 5
 mid, 5
 midl, 6
 midr, 6
 right, 6

linesensors_update
 linesensors.h, 12

MAX_SPEED

motors.h, 14

mid
 linesensors_t, 5

midl
 linesensors_t, 6

midr
 linesensors_t, 6

motor_left
 motors.h, 15

motor_right
 motors.h, 15

motor_t, 6
 direction, 6
 speed, 6

motors.h
 MAX_SPEED, 14
 motor_left, 15
 motor_right, 15
 motors_forward, 14
 motors_init, 14
 motors_lean_neg, 14
 motors_lean_pos, 14
 motors_reverse, 15
 motors_rotate_neg, 15
 motors_rotate_pos, 15
 motors_stop, 15
 motors_update, 15
 ROTATE_SPEED, 14

motors_forward
 motors.h, 14

motors_init
 motors.h, 14

motors_lean_neg
 motors.h, 14

motors_lean_pos
 motors.h, 14

motors_reverse
 motors.h, 15

motors_rotate_neg
 motors.h, 15

motors_rotate_pos
 motors.h, 15

motors_stop
 motors.h, 15

motors_update
 motors.h, 15

ROTATE_SPEED
 motors.h, 14

read

- uart_buffer, 7
- right
 - linesensors_t, 6
- speed
 - motor_t, 6
- UART_BAUDRATE
 - config.h, 9
- UART_BUFFER_SIZE
 - config.h, 9
- UART_DEMULT
 - uart.h, 16
- uart.h
 - UART_DEMULT, 16
 - uart_available, 16
 - uart_find, 17
 - uart_getc, 17
 - uart_gets, 17
 - uart_init, 17
 - uart_putc, 17
 - uart_puts, 17
 - uart_read, 17
 - uart_read_until, 18
 - uart_readline, 18
 - uart_write, 18
- uart_available
 - uart.h, 16
- uart_buffer, 7
 - buffer, 7
 - read, 7
 - write, 7
- uart_find
 - uart.h, 17
- uart_getc
 - uart.h, 17
- uart_gets
 - uart.h, 17
- uart_init
 - uart.h, 17
- uart_putc
 - uart.h, 17
- uart_puts
 - uart.h, 17
- uart_read
 - uart.h, 17
- uart_read_until
 - uart.h, 18
- uart_readline
 - uart.h, 18
- uart_write
 - uart.h, 18
- write
 - uart_buffer, 7