

libdorobo32

1.0.0

Generated by Doxygen 1.8.9.1

Fri May 6 2016 11:28:24



# Contents

<b>1</b>	<b>Todo List</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	motor_t Struct Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Field Documentation . . . . .	7
4.1.2.1	controlPIN1 . . . . .	7
4.1.2.2	controlPIN2 . . . . .	7
4.1.2.3	controlPORT1 . . . . .	7
4.1.2.4	controlPORT2 . . . . .	7
4.1.2.5	speed . . . . .	7
4.1.2.6	timerChannel . . . . .	8
4.2	pin_t Struct Reference . . . . .	8
4.2.1	Detailed Description . . . . .	8
4.2.2	Field Documentation . . . . .	8
4.2.2.1	pin . . . . .	8
4.2.2.2	port . . . . .	8
<b>5</b>	<b>File Documentation</b>	<b>9</b>
5.1	dorobo32/include/adc.h File Reference . . . . .	9
5.1.1	Detailed Description . . . . .	10
5.1.2	Enumeration Type Documentation . . . . .	10
5.1.2.1	DA_ADC_CHANNEL_E . . . . .	10
5.1.3	Function Documentation . . . . .	10
5.1.3.1	adc_get_value . . . . .	10
5.1.3.2	adc_init . . . . .	11
5.2	dorobo32/include/digital.h File Reference . . . . .	11

5.2.1	Detailed Description	12
5.2.2	Enumeration Type Documentation	12
5.2.2.1	DD_DIPS_E	12
5.2.2.2	DD_PINCONFIG_E	12
5.2.2.3	DD_PINLEVEL_E	13
5.2.2.4	DD_PINS_E	13
5.2.3	Function Documentation	13
5.2.3.1	digital_configure_pin	13
5.2.3.2	digital_get_dip	14
5.2.3.3	digital_get_pin	14
5.2.3.4	digital_init	14
5.2.3.5	digital_set_pin	15
5.3	dorobo32/include/dorobo32.h File Reference	15
5.3.1	Detailed Description	16
5.3.2	Function Documentation	16
5.3.2.1	dorobo_init	16
5.3.2.2	led_green	17
5.3.2.3	led_red	17
5.4	dorobo32/include/fft.h File Reference	17
5.4.1	Detailed Description	18
5.4.2	Enumeration Type Documentation	18
5.4.2.1	DFFT_FFT_FREQ_E	18
5.4.3	Function Documentation	18
5.4.3.1	fft_init	18
5.4.3.2	fft_simple	18
5.5	dorobo32/include/motor.h File Reference	19
5.5.1	Detailed Description	19
5.5.2	Enumeration Type Documentation	20
5.5.2.1	DM_MOTORS_E	20
5.5.3	Function Documentation	20
5.5.3.1	motor_init	20
5.5.3.2	motor_set	20
5.6	dorobo32/include/uart.h File Reference	20
5.6.1	Detailed Description	21
5.6.2	Enumeration Type Documentation	21
5.6.2.1	DUART_UART_E	21
5.6.3	Function Documentation	22
5.6.3.1	uart_init	22
5.6.3.2	uart_receive	22
5.6.3.3	uart_send	22

5.6.3.4	uart_send_buffer . . . . .	22
5.7	dorobo32/include/wlan.h File Reference . . . . .	23
5.7.1	Detailed Description . . . . .	23
5.7.2	Function Documentation . . . . .	23
5.7.2.1	wlan_init . . . . .	23
5.8	dorobo32/src/adc.c File Reference . . . . .	23
5.8.1	Function Documentation . . . . .	24
5.8.1.1	adc_get_value . . . . .	24
5.8.2	Variable Documentation . . . . .	24
5.8.2.1	hadc . . . . .	24
5.8.2.2	sConfig . . . . .	25
5.9	dorobo32/src/digital.c File Reference . . . . .	25
5.9.1	Function Documentation . . . . .	25
5.9.1.1	digital_configure_pin . . . . .	25
5.9.1.2	digital_get_dip . . . . .	26
5.9.1.3	digital_get_pin . . . . .	26
5.9.1.4	digital_init . . . . .	26
5.9.1.5	digital_set_pin . . . . .	27
5.10	dorobo32/src/dorobo32.c File Reference . . . . .	27
5.10.1	Function Documentation . . . . .	28
5.10.1.1	dorobo_init . . . . .	28
5.10.1.2	init . . . . .	28
5.10.1.3	led_green . . . . .	29
5.10.1.4	led_red . . . . .	30
5.11	dorobo32/src/fft.c File Reference . . . . .	30
5.11.1	Macro Definition Documentation . . . . .	31
5.11.1.1	SAMPLES . . . . .	31
5.11.2	Function Documentation . . . . .	31
5.11.2.1	fft_get_samples . . . . .	31
5.11.2.2	fft_simple . . . . .	31
5.11.2.3	TIM6_DAC_IRQHandler . . . . .	32
5.11.3	Variable Documentation . . . . .	32
5.11.3.1	fft_sample_index . . . . .	32
5.11.3.2	fft_samples_ready . . . . .	32
5.11.3.3	htim6 . . . . .	32
5.11.3.4	sample_pin . . . . .	32
5.12	dorobo32/src/motor.c File Reference . . . . .	32
5.12.1	Function Documentation . . . . .	33
5.12.1.1	motor_set . . . . .	33
5.12.2	Variable Documentation . . . . .	33

5.12.2.1	htim3	33
5.12.2.2	sConfigOC	34
5.13	dorobo32/src/uart.c File Reference	34
5.13.1	Function Documentation	35
5.13.1.1	uart_receive	35
5.13.1.2	uart_send	36
5.13.1.3	uart_send_buffer	36
5.13.1.4	USART1_IRQHandler	36
5.13.1.5	USART2_IRQHandler	36
5.13.2	Variable Documentation	36
5.13.2.1	huart1	36
5.13.2.2	huart2	36
5.14	dorobo32/src/wlan.c File Reference	36
5.14.1	Function Documentation	37
5.14.1.1	wlan_init	37
<b>Index</b>		<b>39</b>

# Chapter 1

## Todo List

**globalScope> Global `adc_init` (void)**

Please implement me!

**globalScope> Global `fft_init` (void)**

Please implement me!

**globalScope> Global `motor_init` (void)**

Please implement me!

**globalScope> Global `uart_init` (void)**

Please implement me!





## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">motor_t</a>	.....	7
<a href="#">pin_t</a>	.....	8



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

dorobo32/include/ <a href="#">adc.h</a> . . . . .	9
dorobo32/include/ <a href="#">digital.h</a> . . . . .	11
dorobo32/include/ <a href="#">dorobo32.h</a> . . . . .	15
dorobo32/include/ <a href="#">fft.h</a> . . . . .	17
dorobo32/include/ <a href="#">motor.h</a> . . . . .	19
dorobo32/include/ <a href="#">uart.h</a> . . . . .	20
dorobo32/include/ <a href="#">wlan.h</a> . . . . .	23
dorobo32/src/ <a href="#">adc.c</a> . . . . .	23
dorobo32/src/ <a href="#">digital.c</a> . . . . .	25
dorobo32/src/ <a href="#">dorobo32.c</a> . . . . .	27
dorobo32/src/ <a href="#">fft.c</a> . . . . .	30
dorobo32/src/ <a href="#">motor.c</a> . . . . .	32
dorobo32/src/ <a href="#">uart.c</a> . . . . .	34
dorobo32/src/ <a href="#">wlan.c</a> . . . . .	36



## Chapter 4

# Data Structure Documentation

### 4.1 motor\_t Struct Reference

#### Data Fields

- GPIO\_TypeDef \* [controlPORT1](#)
- uint16\_t [controlPIN1](#)
- GPIO\_TypeDef \* [controlPORT2](#)
- uint16\_t [controlPIN2](#)
- uint32\_t [timerChannel](#)
- int8\_t [speed](#)

#### 4.1.1 Detailed Description

Definition at line 18 of file motor.c.

#### 4.1.2 Field Documentation

##### 4.1.2.1 uint16\_t controlPIN1

Definition at line 21 of file motor.c.

##### 4.1.2.2 uint16\_t controlPIN2

Definition at line 23 of file motor.c.

##### 4.1.2.3 GPIO\_TypeDef\* controlPORT1

Definition at line 20 of file motor.c.

##### 4.1.2.4 GPIO\_TypeDef\* controlPORT2

Definition at line 22 of file motor.c.

##### 4.1.2.5 int8\_t speed

Definition at line 25 of file motor.c.

#### 4.1.2.6 uint32\_t timerChannel

Definition at line 24 of file motor.c.

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

- dorobo32/src/[motor.c](#)

## 4.2 pin\_t Struct Reference

### Data Fields

- GPIO\_TypeDef \* [port](#)
- uint16\_t [pin](#)

### 4.2.1 Detailed Description

DoroboLib32 Digital IO

Funktionen für die Manipulation der digitalen IO Pins.

Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

Definition at line 15 of file digital.c.

### 4.2.2 Field Documentation

#### 4.2.2.1 uint16\_t pin

Definition at line 18 of file digital.c.

#### 4.2.2.2 GPIO\_TypeDef\* port

Definition at line 17 of file digital.c.

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

- dorobo32/src/[digital.c](#)

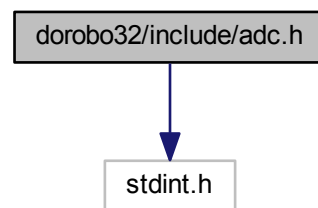
## Chapter 5

# File Documentation

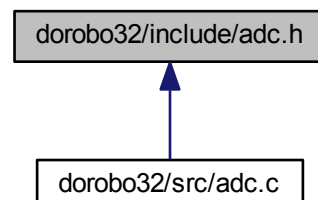
### 5.1 dorobo32/include/adc.h File Reference

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

Include dependency graph for adc.h:



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



## Enumerations

- enum [DA\\_ADC\\_CHANNEL\\_E](#) {  
[DA\\_ADC\\_CHANNEL0](#), [DA\\_ADC\\_CHANNEL1](#), [DA\\_ADC\\_CHANNEL2](#), [DA\\_ADC\\_CHANNEL3](#),  
[DA\\_ADC\\_CHANNEL4](#), [DA\\_ADC\\_CHANNEL5](#), [DA\\_ADC\\_CHANNEL6](#), [DA\\_ADC\\_CHANNEL7](#),  
[DA\\_ADC\\_CHANNEL8](#), [DA\\_ADC\\_CHANNEL9](#) }

## Functions

- void [adc\\_init](#) (void)  
*Adc initialization.*
- uint32\_t [adc\\_get\\_value](#) (enum [DA\\_ADC\\_CHANNEL\\_E](#) adc\_channel\_no)  
*Get converted analog value from analog pin adc\_channel\_no.*

### 5.1.1 Detailed Description

DoroboLib32 ADC (DA)

Functions for reading analog voltages using the analog digital converter.

Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

### 5.1.2 Enumeration Type Documentation

#### 5.1.2.1 enum [DA\\_ADC\\_CHANNEL\\_E](#)

Dorobo32 analog input channels

#### Enumerator

**[DA\\_ADC\\_CHANNEL0](#)** analog channel 0  
**[DA\\_ADC\\_CHANNEL1](#)** analog channel 1  
**[DA\\_ADC\\_CHANNEL2](#)** analog channel 2  
**[DA\\_ADC\\_CHANNEL3](#)** analog channel 3  
**[DA\\_ADC\\_CHANNEL4](#)** analog channel 4  
**[DA\\_ADC\\_CHANNEL5](#)** analog channel 5  
**[DA\\_ADC\\_CHANNEL6](#)** analog channel 6  
**[DA\\_ADC\\_CHANNEL7](#)** analog channel 7  
**[DA\\_ADC\\_CHANNEL8](#)** analog channel 8  
**[DA\\_ADC\\_CHANNEL9](#)** analog channel 9

Definition at line 18 of file adc.h.

### 5.1.3 Function Documentation

#### 5.1.3.1 uint32\_t [adc\\_get\\_value](#) ( enum [DA\\_ADC\\_CHANNEL\\_E](#) adc\_channel\_no )

Get converted analog value from analog pin *adc\_channel\_no*.



## Parameters

<code>adc_channel_no</code>	The adc channel to be read.
-----------------------------	-----------------------------

## Returns

Converted analog 12 bit value.

Definition at line 21 of file adc.c.

## 5.1.3.2 void adc\_init ( void )

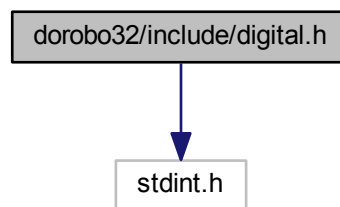
Adc initialization.

**Todo** Please implement me!

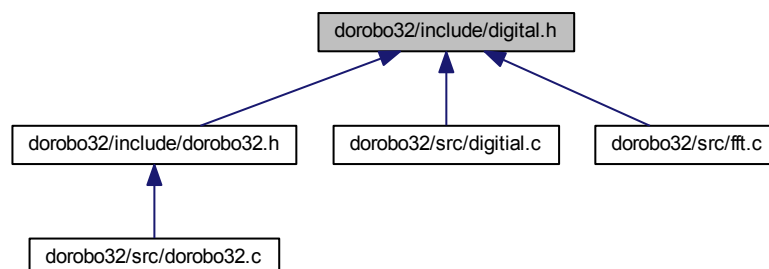
## 5.2 dorobo32/include/digital.h File Reference

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

Include dependency graph for digital.h:



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



## Enumerations

- enum [DD\\_PINS\\_E](#) {  
[DD\\_PIN\\_PD14](#), [DD\\_PIN\\_PD15](#), [DD\\_PIN\\_PC8](#), [DD\\_PIN\\_PC9](#),  
[DD\\_PIN\\_PA8](#), [DD\\_PIN\\_PC13](#), [DD\\_PIN\\_PB11](#), [DD\\_PIN\\_PB10](#),  
[DD\\_PIN\\_PE14](#), [DD\\_PIN\\_PE11](#), [DD\\_PIN\\_PE9](#), [DD\\_PIN\\_PB1](#),  
[DD\\_PIN\\_PF10](#), [DD\\_PIN\\_PF9](#), [DD\\_PIN\\_PE1](#), [DD\\_PIN\\_PE0](#),  
[DD\\_PIN\\_PB3](#), [DD\\_PIN\\_PA15](#) }
- enum [DD\\_DIPS\\_E](#) { [DD\\_DIP1](#), [DD\\_DIP2](#), [DD\\_DIP3](#), [DD\\_DIP4](#) }
- enum [DD\\_PINLEVEL\\_E](#) { [DD\\_LEVEL\\_LOW](#), [DD\\_LEVEL\\_HIGH](#) }
- enum [DD\\_PINCONFIG\\_E](#) { [DD\\_CFG\\_OUTPUT](#), [DD\\_CFG\\_INPUT\\_PULLUP](#), [DD\\_CFG\\_INPUT\\_PULLDOWN](#), [DD\\_CFG\\_INPUT\\_NOPULL](#) }

## Functions

- void [digital\\_init](#) (void)  
Initialize the Dorobo32 Digital I/O (DD) module.
- void [digital\\_configure\\_pin](#) (enum [DD\\_PINS\\_E](#) pin\_no, enum [DD\\_PINCONFIG\\_E](#) direction)  
Configure digital pin direction and pullup/pulldown resistors.
- void [digital\\_set\\_pin](#) (enum [DD\\_PINS\\_E](#) pin\_no, enum [DD\\_PINLEVEL\\_E](#) level)  
Set pin level of pin pin\_no.
- enum [DD\\_PINLEVEL\\_E](#) [digital\\_get\\_pin](#) (enum [DD\\_PINS\\_E](#) pin\_no)  
Get signal level for a pin.
- enum [DD\\_PINLEVEL\\_E](#) [digital\\_get\\_dip](#) (enum [DD\\_DIPS\\_E](#) dip\_no)  
Get position of a dip switch.

### 5.2.1 Detailed Description

DoroboLib32 Digital IO (DD)

Functions to read and set digital IO pins.

Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

### 5.2.2 Enumeration Type Documentation

#### 5.2.2.1 enum [DD\\_DIPS\\_E](#)

Dorobo32 DIP switches

Enumerator

- [DD\\_DIP1](#)** dip switch 1
- [DD\\_DIP2](#)** dip switch 2
- [DD\\_DIP3](#)** dip switch 3
- [DD\\_DIP4](#)** dip switch 4

Definition at line 45 of file digital.h.

#### 5.2.2.2 enum [DD\\_PINCONFIG\\_E](#)

Pin configuration

## Enumerator

**DD\_CFG\_OUTPUT** pin configured as output  
**DD\_CFG\_INPUT\_PULLUP** pin configured as input with internal pullup enabled  
**DD\_CFG\_INPUT\_PULLDOWN** pin configured as input with internal pulldown enabled  
**DD\_CFG\_INPUT\_NOPULL** pin configured as input without internal pullup or pulldown resistor

Definition at line 65 of file digital.h.

## 5.2.2.3 enum DD\_PINLEVEL\_E

Pin signal levels

## Enumerator

**DD\_LEVEL\_LOW** digital low  
**DD\_LEVEL\_HIGH** digital high

Definition at line 56 of file digital.h.

## 5.2.2.4 enum DD\_PINS\_E

Dorobo32 pins available for digital i/o

## Enumerator

**DD\_PIN\_PD14** digital i/o pin D14  
**DD\_PIN\_PD15** digital i/o pin PD15  
**DD\_PIN\_PC8** digital i/o pin PC8  
**DD\_PIN\_PC9** digital i/o pin PC9  
**DD\_PIN\_PA8** digital i/o pin PA8  
**DD\_PIN\_PC13** digital i/o pin PC13  
**DD\_PIN\_PB11** digital i/o pin PB11  
**DD\_PIN\_PB10** digital i/o pin PB10  
**DD\_PIN\_PE14** digital i/o pin PE14  
**DD\_PIN\_PE11** digital i/o pin PE11  
**DD\_PIN\_PE9** digital i/o pin PE9  
**DD\_PIN\_PB1** digital i/o pin PB1  
**DD\_PIN\_PF10** digital i/o pin F10  
**DD\_PIN\_PF9** digital i/o pin F9  
**DD\_PIN\_PE1** digital i/o pin E1  
**DD\_PIN\_PE0** digital i/o pin E0  
**DD\_PIN\_PB3** digital i/o pin B3  
**DD\_PIN\_PA15** digital i/o pin A15

Definition at line 18 of file digital.h.

## 5.2.3 Function Documentation

## 5.2.3.1 void digital\_configure\_pin ( enum DD\_PINS\_E pin\_no, enum DD\_PINCONFIG\_E direction )

Configure digital pin direction and pullup/pulldown resistors.

## Parameters

<i>pin_no</i>	Pin to configure
<i>direction</i>	Predefined configuration as defined in DD_PINCONFIG_E

Definition at line 30 of file digital.c.

### 5.2.3.2 enum DD\_PINLEVEL\_E digital\_get\_dip ( enum DD\_DIPS\_E dip\_no )

Get position of a dip switch.

## Parameters

<i>dip_no</i>	Dip to read from
---------------	------------------

## Returns

Dip signal level as defined in DD\_PINLEVEL\_E

Definition at line 83 of file digital.c.

### 5.2.3.3 enum DD\_PINLEVEL\_E digital\_get\_pin ( enum DD\_PINS\_E pin\_no )

Get signal level for a pin.

## Parameters

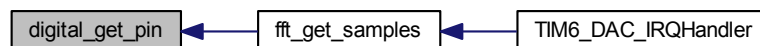
<i>pin_no</i>	Pin to read from
---------------	------------------

## Returns

Pin signal level as defined in DD\_PINLEVEL\_E

Definition at line 76 of file digital.c.

Here is the caller graph for this function:



### 5.2.3.4 void digital\_init ( void )

Initialize the Dorobo32 Digital I/O (DD) module.

Definition at line 25 of file digital.c.

Here is the caller graph for this function:



5.2.3.5 void digital\_set\_pin ( enum DD\_PINS\_E *pin\_no*, enum DD\_PINLEVEL\_E *level* )

Set pin level of pin *pin\_no*.

Parameters

<i>pin_no</i>	Digital pin that is to be set
<i>level</i>	The desired pin level. Values can be DD_LEVEL_LOW or DD_LEVEL_HIGH

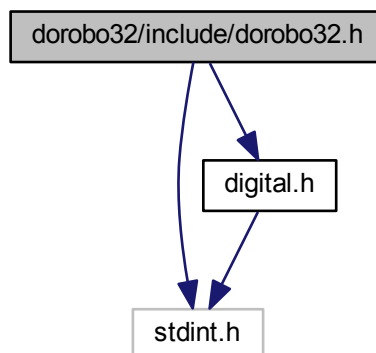
Definition at line 69 of file digital.c.

## 5.3 dorobo32/include/dorobo32.h File Reference

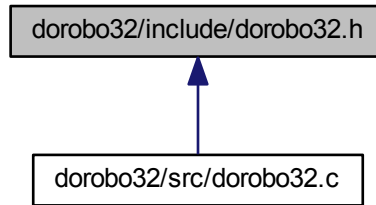
```
#include <stdint.h>
```

```
#include "digital.h"
```

Include dependency graph for dorobo32.h:



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



## Functions

- void [dorobo\\_init](#) (void)

*Board initialization and module initialization for digital i/o and uart.*

- void [led\\_green](#) (enum [DD\\_PINLEVEL\\_E](#) level)

*Set the green on-board led on or off.*

- void [led\\_red](#) (enum [DD\\_PINLEVEL\\_E](#) level)

*Set the red on-board led on or off.*

### 5.3.1 Detailed Description

DoroboLib32 central header

Initialization of the board and the most common modules and control of the on-board LEDs.

Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

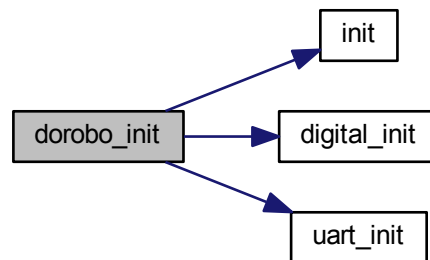
### 5.3.2 Function Documentation

#### 5.3.2.1 void [dorobo\\_init](#) ( void )

Board initialization and module initialization for digital i/o and uart.

Definition at line 19 of file dorobo32.c.

Here is the call graph for this function:



#### 5.3.2.2 void led\_green ( enum DD\_PINLEVEL\_E level )

Set the green on-board led on or off.

Parameters

<i>level</i>	Level as defined in DD_PINLEVEL_E
--------------	-----------------------------------

Definition at line 26 of file dorobo32.c.

#### 5.3.2.3 void led\_red ( enum DD\_PINLEVEL\_E level )

Set the red on-board led on or off.

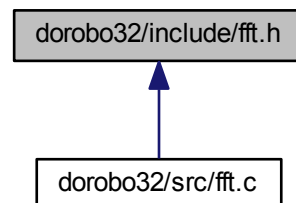
Parameters

<i>level</i>	Level as defined in DD_PINLEVEL_E
--------------	-----------------------------------

Definition at line 31 of file dorobo32.c.

## 5.4 dorobo32/include/fft.h File Reference

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



## Enumerations

- enum `DFFT_FFT_FREQ_E` { `FFT_FREQ100`, `FFT_FREQ125` }

## Functions

- void `fft_init` (void)  
*FFT initialization.*
- uint16\_t `fft_simple` (enum `DFFT_FFT_FREQ_E` `fft_freq`, enum `DD_PINS_E` `pin_no`)  
*Start sampling from digital `pin_no` and perform fft over the samples.*

### 5.4.1 Detailed Description

DoroboLib32 FFT (DFFT)

Simple implementation of a fast fourier transform (fft) to detect 100 Hz and 125 Hz modulated signals.

Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

### 5.4.2 Enumeration Type Documentation

#### 5.4.2.1 enum `DFFT_FFT_FREQ_E`

The signals that can be detected by the function `fft_simple()`

Enumerator

**`FFT_FREQ100`** 100Hz

**`FFT_FREQ125`** 125Hz

Definition at line 18 of file `fft.h`.

### 5.4.3 Function Documentation

#### 5.4.3.1 void `fft_init` ( void )

FFT initialization.

**Todo** Please implement me!

#### 5.4.3.2 uint16\_t `fft_simple` ( enum `DFFT_FFT_FREQ_E` `fft_freq`, enum `DD_PINS_E` `pin_no` )

Start sampling from digital `pin_no` and perform fft over the samples.

Parameters

<code>fft_freq</code>	The frequency that needs to be detected
<code>pin_no</code>	Digital pin that is used for sampling

Returns

The compliance of the measured signal with the signal that is to be detected. A value above 1,200 indicates compliance.

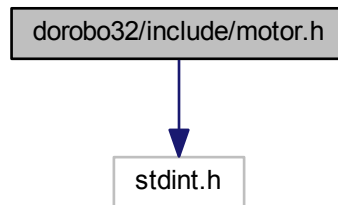
Definition at line 45 of file `fft.c`.



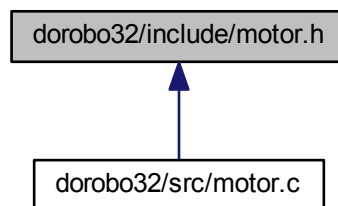
## 5.5 dorobo32/include/motor.h File Reference

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

Include dependency graph for motor.h:



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



### Enumerations

- enum [DM\\_MOTORS\\_E](#) { [DM\\_MOTOR0](#), [DM\\_MOTOR1](#), [DM\\_MOTOR2](#), [DM\\_MOTOR3](#) }

### Functions

- void [motor\\_init](#) (void)  
*Motor initialization.*
- void [motor\\_set](#) (enum [DM\\_MOTORS\\_E](#) motor, int8\_t speed)  
*Set the speed for the given motor.*

#### 5.5.1 Detailed Description

DoroboLib32 Motor (DM)

Functions to set motor speed.

Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

## 5.5.2 Enumeration Type Documentation

### 5.5.2.1 enum `DM_MOTORS_E`

Dorobo32 motors

Enumerator

**`DM_MOTOR0`** Motor 0.

**`DM_MOTOR1`** Motor 1.

**`DM_MOTOR2`** Motor 2.

**`DM_MOTOR3`** Motor 3.

Definition at line 19 of file motor.h.

## 5.5.3 Function Documentation

### 5.5.3.1 void `motor_init` ( void )

Motor initialization.

**Todo** Please implement me!

### 5.5.3.2 void `motor_set` ( enum `DM_MOTORS_E` *motor*, int8\_t *speed* )

Set the speed for the given motor.

Warning: The user of this function is responsible for limiting the motor current by controlling the slope of speed changes.

Parameters

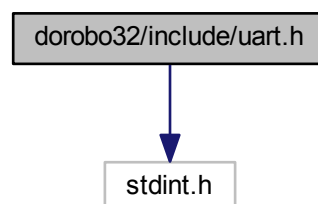
<i>motor</i>	Motor to control
<i>speed</i>	New motor speed and direction in percent (-100-100)

Definition at line 64 of file motor.c.

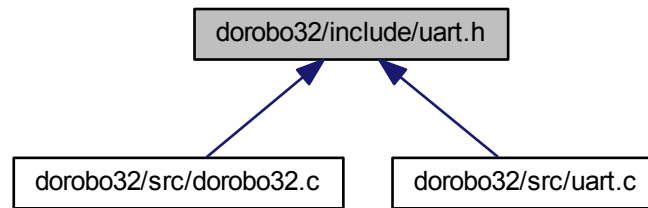
## 5.6 dorobo32/include/uart.h File Reference

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

Include dependency graph for uart.h:



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



## Enumerations

- enum `DUART_UART_E` { `UART1`, `UART2` }

## Functions

- void `uart_init` (void)  
*UART initialization.*
- void `uart_send` (enum `DUART_UART_E` uart, char \*string)  
*Send string via UART.*
- void `uart_send_buffer` (enum `DUART_UART_E` uart, uint8\_t \*pbuffer, uint16\_t size)  
*Send buffer via UART.*
- void `uart_receive` (enum `DUART_UART_E` uart, uint8\_t \*pbuffer, uint16\_t size)  
*Receive messages from UART.*

### 5.6.1 Detailed Description

DoroboLib32 UART (DUART)

Functions for UART communication.

Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

### 5.6.2 Enumeration Type Documentation

#### 5.6.2.1 enum `DUART_UART_E`

Dorobo32 uarts

Enumerator

**`UART1`** UART1 instance.

**`UART2`** UART2 instance.

Definition at line 19 of file `uart.h`.

### 5.6.3 Function Documentation

#### 5.6.3.1 void uart\_init ( void )

UART initialization.

**Todo** Please implement me!

Here is the caller graph for this function:



#### 5.6.3.2 void uart\_receive ( enum DUART\_UART\_E *uart*, uint8\_t \* *pbuffer*, uint16\_t *size* )

Receive messages from UART.

Parameters

<i>uart</i>	UART instance to be used as defined in DUART_UART_E
<i>*pbuffer</i>	Buffer to store received data
<i>size</i>	Number of bytes to be received

Definition at line 34 of file uart.c.

#### 5.6.3.3 void uart\_send ( enum DUART\_UART\_E *uart*, char \* *string* )

Send string via UART.

Parameters

<i>uart</i>	The UART instance to be used as defined in DUART_UART_E
<i>string</i>	char* string to be transmitted

Definition at line 20 of file uart.c.

#### 5.6.3.4 void uart\_send\_buffer ( enum DUART\_UART\_E *uart*, uint8\_t \* *pbuffer*, uint16\_t *size* )

Send buffer via UART.

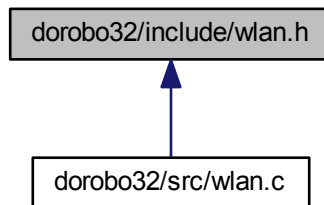
Parameters

<i>uart</i>	The UART instance to be used as defined in DUART_UART_E
<i>*pbuffer</i>	Pointer to buffer
<i>size</i>	Number of character to be sent

Definition at line 27 of file uart.c.

## 5.7 dorobo32/include/wlan.h File Reference

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



### Functions

- void `wlan_init` (void)

*Wlan initialization.*

#### 5.7.1 Detailed Description

DoroboLib32 WLAN (WI)

Functions for wlan init and std i/o stream redirection.

Copyright (c) 2016 Michael Hoffmann, Claus Fühner

#### 5.7.2 Function Documentation

##### 5.7.2.1 void wlan\_init ( void )

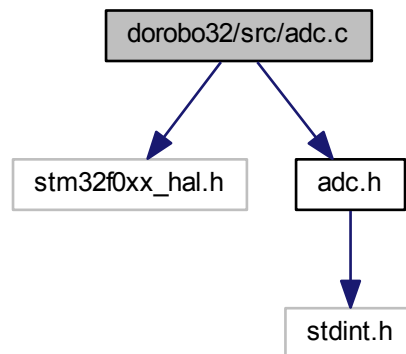
Wlan initialization.

Definition at line 17 of file wlan.c.

## 5.8 dorobo32/src/adc.c File Reference

```
#include "stm32f0xx_hal.h"
#include "adc.h"
```

Include dependency graph for `adc.c`:



## Functions

- `uint32_t adc_get_value` (enum `DA_ADC_CHANNEL_E` `adc_channel_no`)  
*Get converted analog value from analog pin `adc_channel_no`.*

## Variables

- `ADC_HandleTypeDef` `hadc`
- `ADC_ChannelConfTypeDef` `sConfig`

### 5.8.1 Function Documentation

#### 5.8.1.1 `uint32_t adc_get_value ( enum DA_ADC_CHANNEL_E adc_channel_no )`

Get converted analog value from analog pin `adc_channel_no`.

##### Parameters

<code>adc_channel_no</code>	The adc channel to be read.
-----------------------------	-----------------------------

##### Returns

Converted analog 12 bit value.

Definition at line 21 of file `adc.c`.

### 5.8.2 Variable Documentation

#### 5.8.2.1 `ADC_HandleTypeDef` `hadc`

DoroboLib32 ADC (DA)

Functions for reading analog voltages using the analog digital converter.

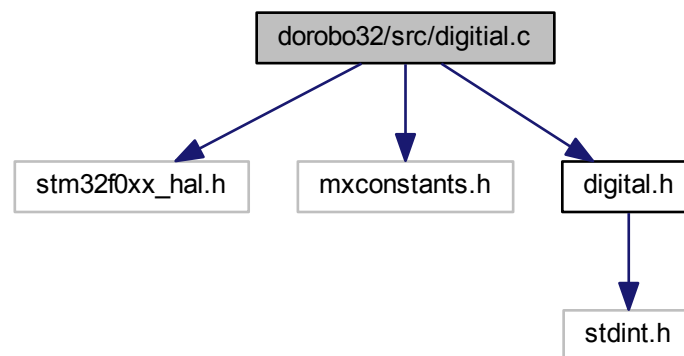
Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

## 5.8.2.2 ADC\_ChannelConfTypeDef sConfig

Definition at line 14 of file adc.c.

## 5.9 dorobo32/src/digital.c File Reference

```
#include "stm32f0xx_hal.h"
#include "mxconstants.h"
#include "digital.h"
Include dependency graph for digital.c:
```



## Data Structures

- struct [pin\\_t](#)

## Functions

- void [digital\\_init](#) (void)  
*Initialize the Dorobo32 Digital I/O (DD) module.*
- void [digital\\_configure\\_pin](#) (enum [DD\\_PINS\\_E](#) pin\_no, enum [DD\\_PINCONFIG\\_E](#) direction)  
*Configure digital pin direction and pullup/pulldown resistors.*
- void [digital\\_set\\_pin](#) (enum [DD\\_PINS\\_E](#) pin\_no, enum [DD\\_PINLEVEL\\_E](#) level)  
*Set pin level of pin pin\_no.*
- enum [DD\\_PINLEVEL\\_E](#) [digital\\_get\\_pin](#) (enum [DD\\_PINS\\_E](#) pin\_no)  
*Get signal level for a pin.*
- enum [DD\\_PINLEVEL\\_E](#) [digital\\_get\\_dip](#) (enum [DD\\_DIPS\\_E](#) dip\_no)  
*Get position of a dip switch.*

## 5.9.1 Function Documentation

5.9.1.1 void digital\_configure\_pin ( enum [DD\\_PINS\\_E](#) pin\_no, enum [DD\\_PINCONFIG\\_E](#) direction )

Configure digital pin direction and pullup/pulldown resistors.

## Parameters

<i>pin_no</i>	Pin to configure
<i>direction</i>	Predefined configuration as defined in DD_PINCONFIG_E

Definition at line 30 of file digital.c.

#### 5.9.1.2 enum DD\_PINLEVEL\_E digital\_get\_dip ( enum DD\_DIPS\_E dip\_no )

Get position of a dip switch.

## Parameters

<i>dip_no</i>	Dip to read from
---------------	------------------

## Returns

Dip signal level as defined in DD\_PINLEVEL\_E

Definition at line 83 of file digital.c.

#### 5.9.1.3 enum DD\_PINLEVEL\_E digital\_get\_pin ( enum DD\_PINS\_E pin\_no )

Get signal level for a pin.

## Parameters

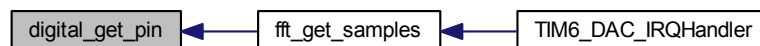
<i>pin_no</i>	Pin to read from
---------------	------------------

## Returns

Pin signal level as defined in DD\_PINLEVEL\_E

Definition at line 76 of file digital.c.

Here is the caller graph for this function:



#### 5.9.1.4 void digital\_init ( void )

Initialize the Dorobo32 Digital I/O (DD) module.

Definition at line 25 of file digital.c.



Here is the caller graph for this function:



#### 5.9.1.5 void digital\_set\_pin ( enum DD\_PINS\_E pin\_no, enum DD\_PINLEVEL\_E level )

Set pin level of pin *pin\_no*.

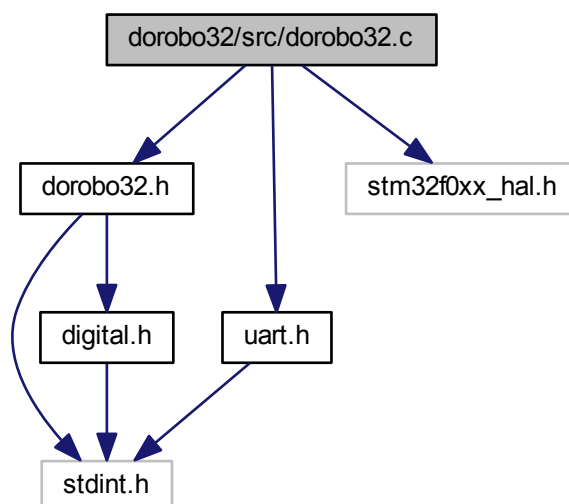
##### Parameters

<i>pin_no</i>	Digital pin that is to be set
<i>level</i>	The desired pin level. Values can be DD_LEVEL_LOW or DD_LEVEL_HIGH

Definition at line 69 of file `digital.c`.

## 5.10 dorobo32/src/dorobo32.c File Reference

```
#include "dorobo32.h"
#include "uart.h"
#include "stm32f0xx_hal.h"
Include dependency graph for dorobo32.c:
```



## Functions

- void `init` ()
- void `dorobo_init` (void)  
*Board initialization and module initialization for digital i/o and uart.*
- void `led_green` (enum `DD_PINLEVEL_E` level)  
*Set the green on-board led on or off.*
- void `led_red` (enum `DD_PINLEVEL_E` level)  
*Set the red on-board led on or off.*

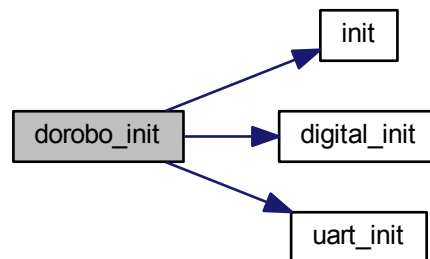
### 5.10.1 Function Documentation

#### 5.10.1.1 void `dorobo_init` ( void )

Board initialization and module initialization for digital i/o and uart.

Definition at line 19 of file `dorobo32.c`.

Here is the call graph for this function:



#### 5.10.1.2 void `init` ( )

DoroboLib32 Sammel-Initialisierung

Initialisierung für die DoroboLib32 und die gebräuchlichsten Module.

Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

Here is the caller graph for this function:



#### 5.10.1.3 void led\_green ( enum DD\_PINLEVEL\_E *level* )

Set the green on-board led on or off.

## Parameters

<i>level</i>	Level as defined in DD_PINLEVEL_E
--------------	-----------------------------------

Definition at line 26 of file dorobo32.c.

#### 5.10.1.4 void led\_red ( enum DD\_PINLEVEL\_E level )

Set the red on-board led on or off.

## Parameters

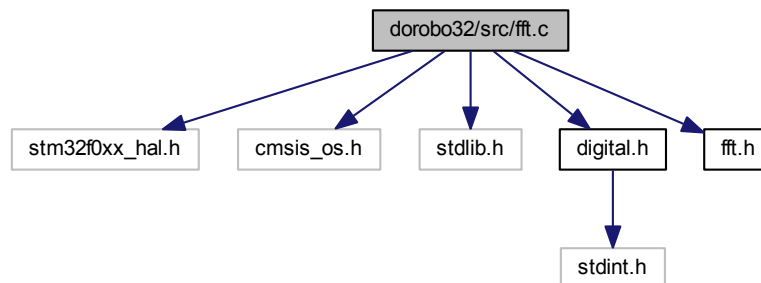
<i>level</i>	Level as defined in DD_PINLEVEL_E
--------------	-----------------------------------

Definition at line 31 of file dorobo32.c.

## 5.11 dorobo32/src/fft.c File Reference

```
#include "stm32f0xx_hal.h"
#include "cmsis_os.h"
#include <stdlib.h>
#include "digital.h"
#include "fft.h"
```

Include dependency graph for fft.c:



## Macros

- `#define SAMPLES 40`

## Functions

- `uint16_t fft_simple (enum DFFT_FFT_FREQ_E efft_freq, enum DD_PINS_E pin_no)`

*Start sampling from digital pin\_no and perform fft over the samples.*

- `void fft_get_samples ()`
- `void TIM6_DAC_IRQHandler (void)`

*This function handles TIM6 global and DAC channel underrun error interrupts.*

## Variables

- TIM\_HandleTypeDef [htim6](#)
- uint8\_t [fft\\_sample\\_index](#) = 0
- uint8\_t [fft\\_samples\\_ready](#) = 0
- enum [DD\\_PINS\\_E](#) [sample\\_pin](#)

## 5.11.1 Macro Definition Documentation

### 5.11.1.1 #define SAMPLES 40

Definition at line 21 of file `fft.c`.

## 5.11.2 Function Documentation

### 5.11.2.1 void `fft_get_samples` ( )

Definition at line 85 of file `fft.c`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.11.2.2 uint16\_t `fft_simple` ( enum `DFFT_FFT_FREQ_E` *fft\_freq*, enum `DD_PINS_E` *pin\_no* )

Start sampling from digital *pin\_no* and perform fft over the samples.

#### Parameters

<i>fft_freq</i>	The frequency that needs to be detected
-----------------	---

<i>pin_no</i>	Digital pin that is used for sampling
---------------	---------------------------------------

#### Returns

The compliance of the measured signal with the signal that is to be detected. A value above 1,200 indicates compliance.

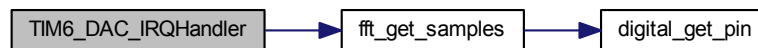
Definition at line 45 of file `fft.c`.

#### 5.11.2.3 void TIM6\_DAC\_IRQHandler ( void )

This function handles TIM6 global and DAC channel underrun error interrupts.

Definition at line 103 of file `fft.c`.

Here is the call graph for this function:



### 5.11.3 Variable Documentation

#### 5.11.3.1 uint8\_t fft\_sample\_index = 0

Definition at line 22 of file `fft.c`.

#### 5.11.3.2 uint8\_t fft\_samples\_ready = 0

Definition at line 23 of file `fft.c`.

#### 5.11.3.3 TIM\_HandleTypeDef htim6

DoroboLib32 FFT (DFFT)

Simple implementation of a fast fourier transform (fft) to detect 100 Hz and 125 Hz modulated signals.

Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

#### 5.11.3.4 enum DD\_PINS\_E sample\_pin

Definition at line 24 of file `fft.c`.

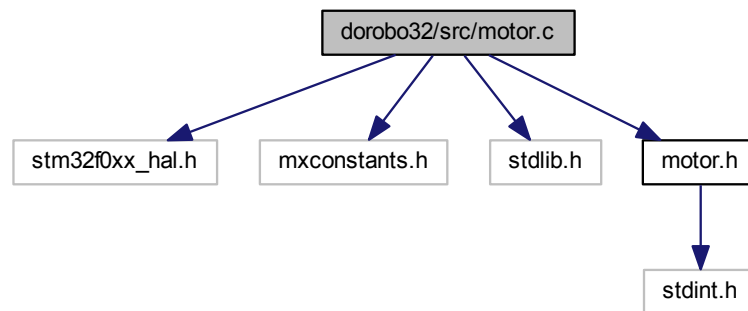
## 5.12 dorobo32/src/motor.c File Reference

```

#include "stm32f0xx_hal.h"
#include "mxconstants.h"
#include <stdlib.h>
#include "motor.h"

```

Include dependency graph for motor.c:



## Data Structures

- struct [motor\\_t](#)

## Functions

- void [motor\\_set](#) (enum [DM\\_MOTORS\\_E](#) motoren, int8\_t speed)  
*Set the speed for the given motor.*

## Variables

- TIM\_HandleTypeDef [htim3](#)
- TIM\_OC\_InitTypeDef [sConfigOC](#)

### 5.12.1 Function Documentation

#### 5.12.1.1 void motor\_set ( enum [DM\\_MOTORS\\_E](#) motor, int8\_t speed )

Set the speed for the given motor.

Warning: The user of this function is responsible for limiting the motor current by controlling the slope of speed changes.

#### Parameters

<i>motor</i>	Motor to control
<i>speed</i>	New motor speed and direction in percent (-100-100)

Definition at line 64 of file motor.c.

### 5.12.2 Variable Documentation

#### 5.12.2.1 TIM\_HandleTypeDef htim3

DoroboLib32 Motor (DM)

Functions to set motor speed.

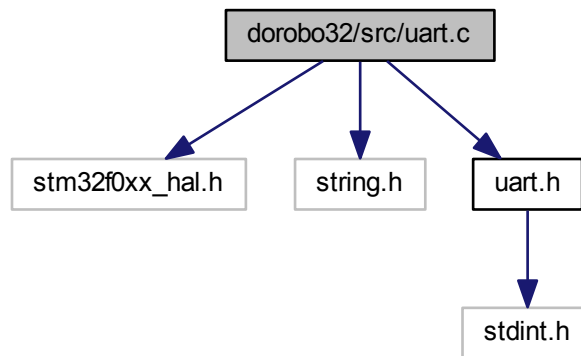
Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

### 5.12.2.2 TIM\_OC\_InitTypeDef sConfigOC

Definition at line 16 of file motor.c.

## 5.13 dorobo32/src/uart.c File Reference

```
#include "stm32f0xx_hal.h"
#include "string.h"
#include "uart.h"
Include dependency graph for uart.c:
```



### Functions

- void `uart_send` (enum `DUART_UART_E` `euart`, char `*msg`)  
*Send string via UART.*
- void `uart_send_buffer` (enum `DUART_UART_E` `uart`, uint8\_t `*pbuffer`, uint16\_t `size`)  
*Send buffer via UART.*
- void `uart_receive` (enum `DUART_UART_E` `euart`, uint8\_t `*pbuffer`, uint16\_t `size`)  
*Receive messages from UART.*
- void `USART1_IRQHandler` (void)
- void `USART2_IRQHandler` (void)

### Variables

- UART\_HandleTypeDef `huart1`
- UART\_HandleTypeDef `huart2`



### 5.13.1 Function Documentation

5.13.1.1 void `uart_receive` ( enum `DUART_UART_E` *uart*, uint8\_t\* *pbuffer*, uint16\_t *size* )

Receive messages from UART.

**Parameters**

<i>uart</i>	UART instance to be used as defined in DUART_UART_E
<i>*pbuffer</i>	Buffer to store received data
<i>size</i>	Number of bytes to be received

Definition at line 34 of file uart.c.

#### 5.13.1.2 void uart\_send ( enum DUART\_UART\_E *uart*, char \* *string* )

Send string via UART.

**Parameters**

<i>uart</i>	The UART instance to be used as defined in DUART_UART_E
<i>string</i>	char* string to be transmitted

Definition at line 20 of file uart.c.

#### 5.13.1.3 void uart\_send\_buffer ( enum DUART\_UART\_E *uart*, uint8\_t \* *pbuffer*, uint16\_t *size* )

Send buffer via UART.

**Parameters**

<i>uart</i>	The UART instance to be used as defined in DUART_UART_E
<i>*pbuffer</i>	Pointer to buffer
<i>size</i>	Number of character to be sent

Definition at line 27 of file uart.c.

#### 5.13.1.4 void USART1\_IRQHandler ( void )

Definition at line 65 of file uart.c.

#### 5.13.1.5 void USART2\_IRQHandler ( void )

Definition at line 70 of file uart.c.

### 5.13.2 Variable Documentation

#### 5.13.2.1 UART\_HandleTypeDef huart1

DoroboLib32 UART (DUART)

Functions for UART communication.

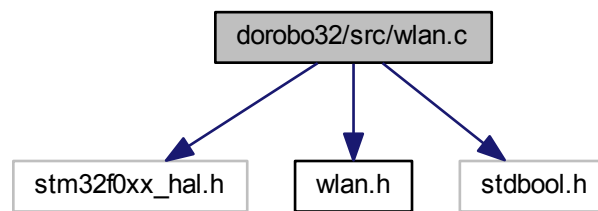
Copyright (c) 2016 Laurent Schröder, Claus Fühner, Michael Hoffmann

#### 5.13.2.2 UART\_HandleTypeDef huart2

## 5.14 dorobo32/src/wlan.c File Reference

```
#include "stm32f0xx_hal.h"
#include "wlan.h"
#include <stdbool.h>
```

Include dependency graph for wlan.c:



## Functions

- void `wlan_init` (void)  
*Wlan initialization.*

### 5.14.1 Function Documentation

#### 5.14.1.1 void wlan\_init ( void )

Wlan initialization.

Definition at line 17 of file `wlan.c`.



# Index

adc.c  
    adc\_get\_value, [24](#)  
    hadc, [24](#)  
    sConfig, [24](#)  
adc.h  
    adc\_get\_value, [10](#)  
    adc\_init, [11](#)  
    DA\_ADC\_CHANNEL0, [10](#)  
    DA\_ADC\_CHANNEL1, [10](#)  
    DA\_ADC\_CHANNEL2, [10](#)  
    DA\_ADC\_CHANNEL3, [10](#)  
    DA\_ADC\_CHANNEL4, [10](#)  
    DA\_ADC\_CHANNEL5, [10](#)  
    DA\_ADC\_CHANNEL6, [10](#)  
    DA\_ADC\_CHANNEL7, [10](#)  
    DA\_ADC\_CHANNEL8, [10](#)  
    DA\_ADC\_CHANNEL9, [10](#)  
    DA\_ADC\_CHANNEL\_E, [10](#)  
adc\_get\_value  
    adc.c, [24](#)  
    adc.h, [10](#)  
adc\_init  
    adc.h, [11](#)  
  
controlPIN1  
    motor\_t, [7](#)  
controlPIN2  
    motor\_t, [7](#)  
controlPORT1  
    motor\_t, [7](#)  
controlPORT2  
    motor\_t, [7](#)  
  
DA\_ADC\_CHANNEL0  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL1  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL2  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL3  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL4  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL5  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL6  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL7  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL8  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL9  
    adc.h, [10](#)  
DA\_ADC\_CHANNEL\_E  
    adc.h, [10](#)  
DD\_CFG\_INPUT\_NOPULL  
    digital.h, [13](#)  
DD\_CFG\_INPUT\_PULLDOWN  
    digital.h, [13](#)  
DD\_CFG\_INPUT\_PULLUP  
    digital.h, [13](#)  
DD\_CFG\_OUTPUT  
    digital.h, [13](#)  
DD\_DIP1  
    digital.h, [12](#)  
DD\_DIP2  
    digital.h, [12](#)  
DD\_DIP3  
    digital.h, [12](#)  
DD\_DIP4  
    digital.h, [12](#)  
DD\_DIPS\_E  
    digital.h, [12](#)  
DD\_LEVEL\_HIGH  
    digital.h, [13](#)  
DD\_LEVEL\_LOW  
    digital.h, [13](#)  
DD\_PIN\_PA15  
    digital.h, [13](#)  
DD\_PIN\_PA8  
    digital.h, [13](#)  
DD\_PIN\_PB1  
    digital.h, [13](#)  
DD\_PIN\_PB10  
    digital.h, [13](#)  
DD\_PIN\_PB11  
    digital.h, [13](#)  
DD\_PIN\_PB3  
    digital.h, [13](#)  
DD\_PIN\_PC13  
    digital.h, [13](#)  
DD\_PIN\_PC8  
    digital.h, [13](#)  
DD\_PIN\_PC9  
    digital.h, [13](#)  
DD\_PIN\_PD14  
    digital.h, [13](#)  
DD\_PIN\_PD15  
    digital.h, [13](#)

DD\_PIN\_PE0  
     digital.h, 13  
 DD\_PIN\_PE1  
     digital.h, 13  
 DD\_PIN\_PE11  
     digital.h, 13  
 DD\_PIN\_PE14  
     digital.h, 13  
 DD\_PIN\_PE9  
     digital.h, 13  
 DD\_PIN\_PF10  
     digital.h, 13  
 DD\_PIN\_PF9  
     digital.h, 13  
 DD\_PINCONFIG\_E  
     digital.h, 12  
 DD\_PINLEVEL\_E  
     digital.h, 13  
 DD\_PINS\_E  
     digital.h, 13  
 DFFT\_FFT\_FREQ\_E  
     fft.h, 18  
 DM\_MOTOR0  
     motor.h, 20  
 DM\_MOTOR1  
     motor.h, 20  
 DM\_MOTOR2  
     motor.h, 20  
 DM\_MOTOR3  
     motor.h, 20  
 DM\_MOTORS\_E  
     motor.h, 20  
 DUART\_UART\_E  
     uart.h, 21  
 digital.h  
     DD\_CFG\_INPUT\_NOPULL, 13  
     DD\_CFG\_INPUT\_PULLDOWN, 13  
     DD\_CFG\_INPUT\_PULLUP, 13  
     DD\_CFG\_OUTPUT, 13  
     DD\_DIP1, 12  
     DD\_DIP2, 12  
     DD\_DIP3, 12  
     DD\_DIP4, 12  
     DD\_DIPS\_E, 12  
     DD\_LEVEL\_HIGH, 13  
     DD\_LEVEL\_LOW, 13  
     DD\_PIN\_PA15, 13  
     DD\_PIN\_PA8, 13  
     DD\_PIN\_PB1, 13  
     DD\_PIN\_PB10, 13  
     DD\_PIN\_PB11, 13  
     DD\_PIN\_PB3, 13  
     DD\_PIN\_PC13, 13  
     DD\_PIN\_PC8, 13  
     DD\_PIN\_PC9, 13  
     DD\_PIN\_PD14, 13  
     DD\_PIN\_PD15, 13  
     DD\_PIN\_PE0, 13  
     DD\_PIN\_PE1, 13  
     DD\_PIN\_PE11, 13  
     DD\_PIN\_PE14, 13  
     DD\_PIN\_PE9, 13  
     DD\_PIN\_PF10, 13  
     DD\_PIN\_PF9, 13  
     DD\_PINCONFIG\_E, 12  
     DD\_PINLEVEL\_E, 13  
     DD\_PINS\_E, 13  
     digital\_configure\_pin, 13  
     digital\_get\_dip, 14  
     digital\_get\_pin, 14  
     digital\_init, 14  
     digital\_set\_pin, 15  
 digital\_configure\_pin  
     digital.h, 13  
     digital.c, 25  
 digital\_get\_dip  
     digital.h, 14  
     digital.c, 26  
 digital\_get\_pin  
     digital.h, 14  
     digital.c, 26  
 digital\_init  
     digital.h, 14  
     digital.c, 26  
 digital\_set\_pin  
     digital.h, 15  
     digital.c, 27  
 digital.c  
     digital\_configure\_pin, 25  
     digital\_get\_dip, 26  
     digital\_get\_pin, 26  
     digital\_init, 26  
     digital\_set\_pin, 27  
 dorobo32.c  
     dorobo\_init, 28  
     init, 28  
     led\_green, 28  
     led\_red, 30  
 dorobo32.h  
     dorobo\_init, 16  
     led\_green, 17  
     led\_red, 17  
 dorobo32/include/adc.h, 9  
 dorobo32/include/digital.h, 11  
 dorobo32/include/dorobo32.h, 15  
 dorobo32/include/fft.h, 17  
 dorobo32/include/motor.h, 19  
 dorobo32/include/uart.h, 20  
 dorobo32/include/wlan.h, 23  
 dorobo32/src/adc.c, 23  
 dorobo32/src/digital.c, 25  
 dorobo32/src/dorobo32.c, 27  
 dorobo32/src/fft.c, 30  
 dorobo32/src/motor.c, 32  
 dorobo32/src/uart.c, 34  
 dorobo32/src/wlan.c, 36

- dorobo\_init
  - dorobo32.c, [28](#)
  - dorobo32.h, [16](#)
- FFT\_FREQ100
  - fft.h, [18](#)
- FFT\_FREQ125
  - fft.h, [18](#)
- fft.c
  - fft\_get\_samples, [31](#)
  - fft\_sample\_index, [32](#)
  - fft\_samples\_ready, [32](#)
  - fft\_simple, [31](#)
  - htim6, [32](#)
  - SAMPLES, [31](#)
  - sample\_pin, [32](#)
  - TIM6\_DAC\_IRQHandler, [32](#)
- fft.h
  - DFFT\_FFT\_FREQ\_E, [18](#)
  - FFT\_FREQ100, [18](#)
  - FFT\_FREQ125, [18](#)
  - fft\_init, [18](#)
  - fft\_simple, [18](#)
- fft\_get\_samples
  - fft.c, [31](#)
- fft\_init
  - fft.h, [18](#)
- fft\_sample\_index
  - fft.c, [32](#)
- fft\_samples\_ready
  - fft.c, [32](#)
- fft\_simple
  - fft.c, [31](#)
  - fft.h, [18](#)
- hadc
  - adc.c, [24](#)
- htim3
  - motor.c, [33](#)
- htim6
  - fft.c, [32](#)
- huart1
  - uart.c, [36](#)
- huart2
  - uart.c, [36](#)
- init
  - dorobo32.c, [28](#)
- led\_green
  - dorobo32.c, [28](#)
  - dorobo32.h, [17](#)
- led\_red
  - dorobo32.c, [30](#)
  - dorobo32.h, [17](#)
- motor.c
  - htim3, [33](#)
  - motor\_set, [33](#)
  - sConfigOC, [34](#)
- motor.h
  - DM\_MOTOR0, [20](#)
  - DM\_MOTOR1, [20](#)
  - DM\_MOTOR2, [20](#)
  - DM\_MOTOR3, [20](#)
  - DM\_MOTORS\_E, [20](#)
  - motor\_init, [20](#)
  - motor\_set, [20](#)
- motor\_init
  - motor.h, [20](#)
- motor\_set
  - motor.c, [33](#)
  - motor.h, [20](#)
- motor\_t, [7](#)
  - controlPIN1, [7](#)
  - controlPIN2, [7](#)
  - controlPORT1, [7](#)
  - controlPORT2, [7](#)
  - speed, [7](#)
  - timerChannel, [7](#)
- pin
  - pin\_t, [8](#)
- pin\_t, [8](#)
  - pin, [8](#)
  - port, [8](#)
- port
  - pin\_t, [8](#)
- SAMPLES
  - fft.c, [31](#)
- sConfig
  - adc.c, [24](#)
- sConfigOC
  - motor.c, [34](#)
- sample\_pin
  - fft.c, [32](#)
- speed
  - motor\_t, [7](#)
- TIM6\_DAC\_IRQHandler
  - fft.c, [32](#)
- timerChannel
  - motor\_t, [7](#)
- UART1
  - uart.h, [21](#)
- UART2
  - uart.h, [21](#)
- USART1\_IRQHandler
  - uart.c, [36](#)
- USART2\_IRQHandler
  - uart.c, [36](#)
- uart.c
  - huart1, [36](#)
  - huart2, [36](#)
  - USART1\_IRQHandler, [36](#)
  - USART2\_IRQHandler, [36](#)

- uart\_receive, [35](#)
- uart\_send, [36](#)
- uart\_send\_buffer, [36](#)
- uart.h
  - DUART\_UART\_E, [21](#)
  - UART1, [21](#)
  - UART2, [21](#)
  - uart\_init, [22](#)
  - uart\_receive, [22](#)
  - uart\_send, [22](#)
  - uart\_send\_buffer, [22](#)
- uart\_init
  - uart.h, [22](#)
- uart\_receive
  - uart.c, [35](#)
  - uart.h, [22](#)
- uart\_send
  - uart.c, [36](#)
  - uart.h, [22](#)
- uart\_send\_buffer
  - uart.c, [36](#)
  - uart.h, [22](#)
- wlan.c
  - wlan\_init, [37](#)
- wlan.h
  - wlan\_init, [23](#)
- wlan\_init
  - wlan.c, [37](#)
  - wlan.h, [23](#)