

lib-arduino

Generated by Doxygen 1.8.15



<b>1 lib-arduino</b>	<b>1</b>
<b>2 Data Structure Index</b>	<b>3</b>
2.1 Data Structures . . . . .	3
<b>3 File Index</b>	<b>5</b>
3.1 File List . . . . .	5
<b>4 Data Structure Documentation</b>	<b>7</b>
4.1 BCD Struct Reference . . . . .	7
4.1.1 Detailed Description . . . . .	7
4.1.2 Field Documentation . . . . .	8
4.1.2.1 a . . . . .	8
4.1.2.2 b . . . . .	8
4.1.2.3 c . . . . .	8
4.1.2.4 d . . . . .	8
4.1.2.5 delay . . . . .	8
4.1.2.6 dp . . . . .	9
4.1.2.7 e . . . . .	9
4.1.2.8 f . . . . .	9
4.1.2.9 g . . . . .	9
4.1.2.10 loop . . . . .	9
4.2 Button Struct Reference . . . . .	9
4.2.1 Detailed Description . . . . .	10
4.2.2 Field Documentation . . . . .	10
4.2.2.1 fn . . . . .	10
4.2.2.2 mode . . . . .	10
4.2.2.3 pin_number . . . . .	10
4.3 bytes Struct Reference . . . . .	11
4.3.1 Field Documentation . . . . .	11
4.3.1.1 COM0 . . . . .	11
4.3.1.2 COM1 . . . . .	11
4.3.1.3 CS0 . . . . .	11
4.3.1.4 CS1 . . . . .	11
4.3.1.5 CS2 . . . . .	12
4.3.1.6 WGM0 . . . . .	12
4.3.1.7 WGM1 . . . . .	12
4.3.1.8 WGM2 . . . . .	12
4.3.1.9 WGM3 . . . . .	12
4.4 Led Struct Reference . . . . .	12
4.4.1 Detailed Description . . . . .	13
4.4.2 Field Documentation . . . . .	13
4.4.2.1 pin_number . . . . .	13

4.4.2.2 state	13
4.5 Pin Struct Reference	13
4.5.1 Detailed Description	14
4.5.2 Field Documentation	14
4.5.2.1 _DDR	14
4.5.2.2 _OCRA	14
4.5.2.3 _OCRB	14
4.5.2.4 _PIN	14
4.5.2.5 _PORT	14
4.5.2.6 _TCCA	15
4.5.2.7 _TCCB	15
4.5.2.8 P	15
<b>5 File Documentation</b>	<b>17</b>
5.1 src/bcd.c File Reference	17
5.1.1 Function Documentation	17
5.1.1.1 bcd_clear()	17
5.1.1.2 bcd_delay()	18
5.1.1.3 bcd_free()	18
5.1.1.4 bcd_set_loop()	18
5.1.1.5 bcd_setup()	19
5.1.1.6 bcd_show_char()	19
5.1.1.7 bcd_show_string()	20
5.2 src/button.c File Reference	20
5.2.1 Function Documentation	20
5.2.1.1 btn_fired()	20
5.2.1.2 btn_pressed()	21
5.2.1.3 btn_register_fn()	21
5.2.1.4 btn_set_mode()	21
5.2.1.5 btn_setup()	22
5.3 src/delay.c File Reference	22
5.3.1 Function Documentation	22
5.3.1.1 delay_1ms()	22
5.3.1.2 delay_1us()	23
5.3.1.3 delay_ms()	23
5.3.1.4 delay_us()	23
5.4 src/include/bcd.h File Reference	23
5.4.1 Function Documentation	24
5.4.1.1 bcd_clear()	24
5.4.1.2 bcd_delay()	24
5.4.1.3 bcd_free()	25
5.4.1.4 bcd_set_loop()	25

5.4.1.5 bcd_setup()	25
5.4.1.6 bcd_show_char()	26
5.4.1.7 bcd_show_string()	26
5.5 src/include/bcd_seven.h File Reference	26
5.5.1 Macro Definition Documentation	29
5.5.1.1 rep_a	30
5.5.1.2 rep_b	30
5.5.1.3 rep_c	30
5.5.1.4 rep_d	30
5.5.1.5 rep_dp	30
5.5.1.6 rep_e	30
5.5.1.7 rep_f	31
5.5.1.8 rep_g	31
5.5.2 Variable Documentation	31
5.5.2.1 SevenSegmentASCII	31
5.5.2.2 sseg_0	31
5.5.2.3 sseg_1	31
5.5.2.4 sseg_2	32
5.5.2.5 sseg_3	32
5.5.2.6 sseg_4	32
5.5.2.7 sseg_5	32
5.5.2.8 sseg_6	32
5.5.2.9 sseg_7	32
5.5.2.10 sseg_8	33
5.5.2.11 sseg_9	33
5.5.2.12 sseg_A	33
5.5.2.13 sseg_a	33
5.5.2.14 sseg_B	33
5.5.2.15 sseg_b	33
5.5.2.16 sseg_C	34
5.5.2.17 sseg_c	34
5.5.2.18 sseg_D	34
5.5.2.19 sseg_d	34
5.5.2.20 sseg_E	34
5.5.2.21 sseg_e	34
5.5.2.22 sseg_F	35
5.5.2.23 sseg_f	35
5.5.2.24 sseg_G	35
5.5.2.25 sseg_g	35
5.5.2.26 sseg_H	35
5.5.2.27 sseg_h	35
5.5.2.28 sseg_I	36

---

5.5.2.29 sseg_i . . . . .	36
5.5.2.30 sseg_J . . . . .	36
5.5.2.31 sseg_j . . . . .	36
5.5.2.32 sseg_K . . . . .	36
5.5.2.33 sseg_k . . . . .	36
5.5.2.34 sseg_L . . . . .	37
5.5.2.35 sseg_l . . . . .	37
5.5.2.36 sseg_M . . . . .	37
5.5.2.37 sseg_m . . . . .	37
5.5.2.38 sseg_N . . . . .	37
5.5.2.39 sseg_n . . . . .	37
5.5.2.40 sseg_O . . . . .	38
5.5.2.41 sseg_o . . . . .	38
5.5.2.42 sseg_P . . . . .	38
5.5.2.43 sseg_p . . . . .	38
5.5.2.44 sseg_Q . . . . .	38
5.5.2.45 sseg_q . . . . .	38
5.5.2.46 sseg_R . . . . .	39
5.5.2.47 sseg_r . . . . .	39
5.5.2.48 sseg_S . . . . .	39
5.5.2.49 sseg_s . . . . .	39
5.5.2.50 sseg_T . . . . .	39
5.5.2.51 sseg_t . . . . .	39
5.5.2.52 sseg_U . . . . .	40
5.5.2.53 sseg_u . . . . .	40
5.5.2.54 sseg_V . . . . .	40
5.5.2.55 sseg_v . . . . .	40
5.5.2.56 sseg_W . . . . .	40
5.5.2.57 sseg_w . . . . .	40
5.5.2.58 sseg_X . . . . .	41
5.5.2.59 sseg_x . . . . .	41
5.5.2.60 sseg_Y . . . . .	41
5.5.2.61 sseg_y . . . . .	41
5.5.2.62 sseg_Z . . . . .	41
5.5.2.63 sseg_z . . . . .	41
5.6 src/include/button.h File Reference . . . . .	42
5.6.1 Typedef Documentation . . . . .	42
5.6.1.1 Button_Mode . . . . .	42
5.6.2 Enumeration Type Documentation . . . . .	42
5.6.2.1 button_mode . . . . .	42
5.6.3 Function Documentation . . . . .	43
5.6.3.1 btn_fired() . . . . .	43

---

5.6.3.2 btn_pressed()	43
5.6.3.3 btn_register_fn()	43
5.6.3.4 btn_set_mode()	44
5.6.3.5 btn_setup()	44
5.7 src/include/delay.h File Reference	44
5.7.1 Function Documentation	45
5.7.1.1 delay_1ms()	45
5.7.1.2 delay_1us()	45
5.7.1.3 delay_ms()	45
5.7.1.4 delay_us()	45
5.8 src/include/led.h File Reference	46
5.8.1 Typedef Documentation	46
5.8.1.1 Led_State	47
5.8.2 Enumeration Type Documentation	47
5.8.2.1 led_state	47
5.8.3 Function Documentation	47
5.8.3.1 led_blink()	47
5.8.3.2 led_free()	47
5.8.3.3 led_kill_light()	48
5.8.3.4 led_make_light()	48
5.8.3.5 led_setup()	48
5.8.3.6 led_switch_light()	49
5.9 src/include/macros.h File Reference	49
5.9.1 Macro Definition Documentation	49
5.9.1.1 clr_bit	49
5.9.1.2 comp_masks	49
5.9.1.3 MACROS	49
5.9.1.4 set_bit	50
5.9.1.5 swt_bit	50
5.9.1.6 tst_bit	50
5.10 src/include/pins.h File Reference	50
5.10.1 Macro Definition Documentation	51
5.10.1.1 F_CPU	51
5.10.1.2 HIGH	51
5.10.1.3 LOW	51
5.10.2 Function Documentation	51
5.10.2.1 clr_pullup()	51
5.10.2.2 digital_write()	52
5.10.2.3 set_input()	52
5.10.2.4 set_output()	52
5.10.2.5 set_pullup()	53
5.10.2.6 test_input()	53

5.10.2.7 toggle_pullup()	53
5.10.3 Variable Documentation	53
5.10.3.1 Bytes	53
5.10.3.2 Pins	54
5.11 src/include/pwm.h File Reference	54
5.11.1 Function Documentation	54
5.11.1.1 lookup_cs()	54
5.11.1.2 lookup_cs2()	55
5.11.1.3 lookup_ocr()	55
5.11.1.4 lookup_prescalar()	55
5.11.1.5 pwmwave()	55
5.11.1.6 sqwave()	56
5.12 src/include/supersonic.h File Reference	56
5.12.1 Function Documentation	56
5.12.1.1 supersonic()	56
5.13 src/include/uart.h File Reference	57
5.13.1 Macro Definition Documentation	57
5.13.1.1 BAUD	57
5.13.1.2 F_CPU	57
5.13.2 Function Documentation	57
5.13.2.1 uart_getchar()	58
5.13.2.2 uart_init()	58
5.13.2.3 uart_putchar()	58
5.14 src/led.c File Reference	58
5.14.1 Function Documentation	58
5.14.1.1 led_blink()	58
5.14.1.2 led_free()	59
5.14.1.3 led_kill_light()	59
5.14.1.4 led_make_light()	59
5.14.1.5 led_setup()	60
5.14.1.6 led_switch_light()	60
5.15 src/pins.c File Reference	60
5.15.1 Function Documentation	61
5.15.1.1 clr_pullup()	61
5.15.1.2 digital_write()	61
5.15.1.3 set_input()	61
5.15.1.4 set_output()	62
5.15.1.5 set_pullup()	62
5.15.1.6 test_input()	62
5.15.1.7 toggle_pullup()	63
5.15.2 Variable Documentation	63
5.15.2.1 Bytes	63



5.15.2.2 Pins . . . . .	63
5.16 src/pwm.c File Reference . . . . .	64
5.16.1 Function Documentation . . . . .	64
5.16.1.1 lookup_cs() . . . . .	64
5.16.1.2 lookup_cs2() . . . . .	64
5.16.1.3 lookup_ocr() . . . . .	65
5.16.1.4 lookup_prescalar() . . . . .	65
5.16.1.5 pwmwave() . . . . .	65
5.16.1.6 sqwave() . . . . .	66
5.17 src/README.md File Reference . . . . .	66
5.18 src/supersonic.c File Reference . . . . .	66
5.18.1 Function Documentation . . . . .	66
5.18.1.1 supersonic() . . . . .	66
5.19 src/uart.c File Reference . . . . .	67
5.19.1 Function Documentation . . . . .	67
5.19.1.1 uart_getchar() . . . . .	67
5.19.1.2 uart_init() . . . . .	67
5.19.1.3 uart_putchar() . . . . .	67
5.19.2 Variable Documentation . . . . .	67
5.19.2.1 uart_io . . . . .	67
<b>Index</b>	<b>69</b>



# Chapter 1

## lib-arduino

Biblioteca para utilização de módulos básicos para Arduino

### Introdução

Biblioteca desenvolvida como primeiro trabalho da disciplina de Sistemas Digitais, curso Ciência da Computação pela Universidade Estadual de Maringá. O trabalho consiste em uma biblioteca para integração com os seguintes módulos para o microcontrolador Arduino:

- Manipulação dos pinos E/S
- Geração de ondas
- Delay variável
- Interface com LED
- Interface com botão
- Interface com Display de 7-segmentos
- Sensor de distância

### Requisitos

A biblioteca foi desenvolvida para ambiente Linux, chip alvo Atmega328p. As dependências:

```
avrduide  
gcc-avr
```

### Utilização

Crie um arquivo fonte *main.c* no diretório *src/exec*, em seguida:

```
make
```

Para utilizar a biblioteca basta incluir as arquivos header. Ex:

```
#include "pins.h"
```

### Documentação

A documentação está disponível em `doc/latex/refman.pdf`.

### Autores

Ricardo Henrique Brunetto ra94182

Thiago Kira ra78750



## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

BCD	7
Button	9
bytes	11
Led	12
Pin	13



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">bcd.c</a>	17
src/ <a href="#">button.c</a>	20
src/ <a href="#">delay.c</a>	22
src/ <a href="#">led.c</a>	58
src/ <a href="#">pins.c</a>	60
src/ <a href="#">pwm.c</a>	64
src/ <a href="#">supersonic.c</a>	66
src/ <a href="#">uart.c</a>	67
src/include/ <a href="#">bcd.h</a>	23
src/include/ <a href="#">bcd_seven.h</a>	26
src/include/ <a href="#">button.h</a>	42
src/include/ <a href="#">delay.h</a>	44
src/include/ <a href="#">led.h</a>	46
src/include/ <a href="#">macros.h</a>	49
src/include/ <a href="#">pins.h</a>	50
src/include/ <a href="#">pwm.h</a>	54
src/include/ <a href="#">supersonic.h</a>	56
src/include/ <a href="#">uart.h</a>	57





## Chapter 4

# Data Structure Documentation

### 4.1 BCD Struct Reference

```
#include <bcd.h>
```

#### Data Fields

- `uint8_t a`  
*Represents the display pin 'a'.*
- `uint8_t b`  
*Represents the display pin 'b'.*
- `uint8_t c`  
*Represents the display pin 'c'.*
- `uint8_t d`  
*Represents the display pin 'd'.*
- `uint8_t e`  
*Represents the display pin 'e'.*
- `uint8_t f`  
*Represents the display pin 'f'.*
- `uint8_t g`  
*Represents the display pin 'g'.*
- `uint8_t dp`  
*Represents the display pin 'dot'.*
- `double delay`  
*Represents a delay between character transition.*
- `uint8_t loop`  
*Stores the current display mode: 0 for unique exhibition, or any other value for cyclic exhibition.*

#### 4.1.1 Detailed Description

Standard 7-seg Display structure

**BCD** implements a simple display that can be used by setting up each digital pin where display is connected. This structure contains the main functions while using a 7-seg display, providing a simple, but powerful, interface to be expanded.

## 4.1.2 Field Documentation

### 4.1.2.1 a

```
uint8_t BCD::a
```

Represents the display pin 'a'.

### 4.1.2.2 b

```
uint8_t BCD::b
```

Represents the display pin 'b'.

### 4.1.2.3 c

```
uint8_t BCD::c
```

Represents the display pin 'c'.

### 4.1.2.4 d

```
uint8_t BCD::d
```

Represents the display pin 'd'.

### 4.1.2.5 delay

```
double BCD::delay
```

Represents a delay between character transition.

#### 4.1.2.6 dp

```
uint8_t BCD::dp
```

Represents the display pin 'dot'.

#### 4.1.2.7 e

```
uint8_t BCD::e
```

Represents the display pin 'e'.

#### 4.1.2.8 f

```
uint8_t BCD::f
```

Represents the display pin 'f'.

#### 4.1.2.9 g

```
uint8_t BCD::g
```

Represents the display pin 'g'.

#### 4.1.2.10 loop

```
uint8_t BCD::loop
```

Stores the current display mode: 0 for unique exhibition, or any other value for cyclic exhibition.

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

- [src/include/bcd.h](#)

## 4.2 Button Struct Reference

```
#include <button.h>
```

## Data Fields

- `uint8_t pin_number`  
*Digital pin number associated to [Button](#).*
- `Button_Mode mode`  
*Specifies if the button waits to be released to fire action.*
- `void(* fn )()`  
*Registered function, to be called whether button is fired.*

### 4.2.1 Detailed Description

Standard [Button](#) Structure

This file provides a simple [Button](#) Structure, with a simple interface to basic operations involving buttons.

### 4.2.2 Field Documentation

#### 4.2.2.1 fn

```
void(* Button::fn) ()
```

Registered function, to be called whether button is fired.

#### 4.2.2.2 mode

```
Button_Mode Button::mode
```

Specifies if the button waits to be released to fire action.

#### 4.2.2.3 pin\_number

```
uint8_t Button::pin_number
```

Digital pin number associated to [Button](#).

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

- `src/include/button.h`

## 4.3 bytes Struct Reference

```
#include <pins.h>
```

### Data Fields

- volatile uint8\_t [WGM0](#)
- volatile uint8\_t [WGM1](#)
- volatile uint8\_t [WGM2](#)
- volatile uint8\_t [COM0](#)
- volatile uint8\_t [COM1](#)
- volatile uint8\_t [CS0](#)
- volatile uint8\_t [CS1](#)
- volatile uint8\_t [CS2](#)
- volatile uint8\_t [WGM3](#)

### 4.3.1 Field Documentation

#### 4.3.1.1 COM0

```
volatile uint8_t bytes::COM0
```

#### 4.3.1.2 COM1

```
volatile uint8_t bytes::COM1
```

#### 4.3.1.3 CS0

```
volatile uint8_t bytes::CS0
```

#### 4.3.1.4 CS1

```
volatile uint8_t bytes::CS1
```

#### 4.3.1.5 CS2

```
volatile uint8_t bytes::CS2
```

#### 4.3.1.6 WGM0

```
volatile uint8_t bytes::WGM0
```

#### 4.3.1.7 WGM1

```
volatile uint8_t bytes::WGM1
```

#### 4.3.1.8 WGM2

```
volatile uint8_t bytes::WGM2
```

#### 4.3.1.9 WGM3

```
volatile uint8_t bytes::WGM3
```

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

- [src/include/pins.h](#)

## 4.4 Led Struct Reference

```
#include <led.h>
```

### Data Fields

- int [pin\\_number](#)  
*Digital pin number associated to [Led](#).*
- [Led\\_State](#) state  
*Represents a [Led](#) State, used to control on/off functions.*

### 4.4.1 Detailed Description

Standard [Led](#) Structure

This file provides a simple [Led](#) Structure, with a simple interface to basic operations.

### 4.4.2 Field Documentation

#### 4.4.2.1 `pin_number`

```
int Led::pin_number
```

Digital pin number associated to [Led](#).

#### 4.4.2.2 `state`

```
Led_State Led::state
```

Represents a [Led](#) State, used to control on/off functions.

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

- `src/include/led.h`

## 4.5 Pin Struct Reference

```
#include <pins.h>
```

### Data Fields

- `volatile uint8_t P`  
*Pin number associated by ATmega328p.*
- `volatile uint8_t * _PORT`  
*Points to a PORT register (pullup)*
- `volatile uint8_t * _DDR`  
*Points to a DDR register (data direction)*
- `volatile uint8_t * _PIN`  
*Points to a PIN register (input signal)*
- `volatile uint8_t * _TCCA`  
*Points to the TCCA register (PWM)*
- `volatile uint8_t * _TCCB`  
*Points to the TCCB register (PWM)*
- `volatile uint8_t * _OCRA`  
*Points to the OCRA register (PWM)*
- `volatile uint8_t * _OCRB`  
*Points to the OCRB register (PWM)*

### 4.5.1 Detailed Description

Standard [Pin](#) Structure

This file provides a simple [Pin](#) Structure, which contains sufficient data to allow the basic operations that involves I/O.

### 4.5.2 Field Documentation

#### 4.5.2.1 \_DDR

```
volatile uint8_t* Pin::_DDR
```

Points to a DDR register (data direction)

#### 4.5.2.2 \_OCRA

```
volatile uint8_t* Pin::_OCRA
```

Points to the OCRA register (PWM)

#### 4.5.2.3 \_OCRB

```
volatile uint8_t* Pin::_OCRB
```

Points to the OCRB register (PWM)

#### 4.5.2.4 \_PIN

```
volatile uint8_t* Pin::_PIN
```

Points to a PIN register (input signal)

#### 4.5.2.5 \_PORT

```
volatile uint8_t* Pin::_PORT
```

Points to a PORT register (pullup)



#### 4.5.2.6 `_TCCA`

```
volatile uint8_t* Pin::_TCCA
```

Points to the TCCA register (PWM)

#### 4.5.2.7 `_TCCB`

```
volatile uint8_t* Pin::_TCCB
```

Points to the TCCB register (PWM)

#### 4.5.2.8 `P`

```
volatile uint8_t Pin::P
```

[Pin](#) number associated by ATmega328p.

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

- [src/include/pins.h](#)



## Chapter 5

# File Documentation

### 5.1 src/bcd.c File Reference

```
#include "bcd.h"  
#include "bcd_seven.h"
```

#### Functions

- **BCD \* bcd\_setup** (uint8\_t a, uint8\_t b, uint8\_t c, uint8\_t d, uint8\_t e, uint8\_t f, uint8\_t g, uint8\_t dp, double delay)  
*Setups a new instance of a display.*
- void **bcd\_show\_char** (BCD \*display, char c)  
*Displays an ASCII character in a display.*
- void **bcd\_set\_loop** (BCD \*display, uint8\_t loop)  
*Sets a loop mode for a display.*
- void **bcd\_delay** (BCD \*display)  
*Freezes a display for a presetted delay.*
- void **bcd\_clear** (BCD \*display)  
*Clears a display.*
- void **bcd\_show\_string** (BCD \*display, char \*str)  
*Displays a string in a display, a character at a time, interleaved by a delay.*
- void **bcd\_free** (BCD \*display)  
*Destroys a display instance.*

#### 5.1.1 Function Documentation

##### 5.1.1.1 bcd\_clear()

```
void bcd_clear (  
    BCD * display )
```

Clears a display.

**Parameters**

<i>display</i>	Display instance.
----------------	-------------------

**5.1.1.2 bcd\_delay()**

```
void bcd_delay (
    BCD * display )
```

Freezes a display for a presetted delay.

**Parameters**

<i>display</i>	Display instance.
----------------	-------------------

**5.1.1.3 bcd\_free()**

```
void bcd_free (
    BCD * display )
```

Destroys a display instance.

**Parameters**

<i>display</i>	Display instance.
----------------	-------------------

**5.1.1.4 bcd\_set\_loop()**

```
void bcd_set_loop (
    BCD * display,
    uint8_t loop )
```

Sets a loop mode for a display.

**Parameters**

<i>display</i>	Display instance.
<i>loop</i>	Defines a unique exhibition (0) or a cyclic exhibition (any other value).

## 5.1.1.5 bcd\_setup()

```
BCD* bcd_setup (
    uint8_t a,
    uint8_t b,
    uint8_t c,
    uint8_t d,
    uint8_t e,
    uint8_t f,
    uint8_t g,
    uint8_t dp,
    double delay )
```

Setups a new instance of a display.

## Parameters

<i>a</i>	Digital pin where the display pin 'a' is plugged.
<i>b</i>	Digital pin where the display pin 'b' is plugged.
<i>c</i>	Digital pin where the display pin 'c' is plugged.
<i>d</i>	Digital pin where the display pin 'd' is plugged.
<i>e</i>	Digital pin where the display pin 'e' is plugged.
<i>f</i>	Digital pin where the display pin 'f' is plugged.
<i>g</i>	Digital pin where the display pin 'g' is plugged.
<i>dp</i>	Digital pin where the dot pin of display is plugged.
<i>delay</i>	Delay between transactions, if multiple characters are going to be shown.

## Returns

A pointer to display structure.

## 5.1.1.6 bcd\_show\_char()

```
void bcd_show_char (
    BCD * display,
    char character )
```

Displays an ASCII character in a display.

## Parameters

<i>display</i>	Display instance.
<i>character</i>	Character to be shown.

### 5.1.1.7 bcd\_show\_string()

```
void bcd_show_string (
    BCD * display,
    char * str )
```

Displays a string in a display, a character at a time, interleaved by a delay.

#### Parameters

<i>display</i>	Display instance.
<i>str</i>	String to be shown.

## 5.2 src/button.c File Reference

```
#include "button.h"
#include <stdlib.h>
```

### Functions

- [Button](#) \* [btn\\_setup](#) (uint8\_t pin\_number)  
*Setups a new instance of a [Button](#).*
- void [btn\\_register\\_fn](#) ([Button](#) \*b, void \*function)  
*Registers a function to be called whether button is fired.*
- void [btn\\_set\\_mode](#) ([Button](#) \*b, [Button\\_Mode](#) mode)  
*Changes button behavior, waiting or not to be released to fire action.*
- void [btn\\_fired](#) ([Button](#) \*b)  
*Verifies if a button is pressed at moment and fires it's action.*
- uint8\_t [btn\\_pressed](#) ([Button](#) \*b)  
*Verifies if a button is pressed at moment.*

### 5.2.1 Function Documentation

#### 5.2.1.1 btn\_fired()

```
void btn_fired (
    Button * b )
```

Verifies if a button is pressed at moment and fires it's action.

#### Parameters

<i>b</i>	<a href="#">Button</a> instance.
----------	----------------------------------

### 5.2.1.2 btn\_pressed()

```
uint8_t btn_pressed (
    Button * b )
```

Verifies if a button is pressed at moment.

#### Parameters

<i>b</i>	Button instance.
----------	------------------

#### Returns

Returns 0 if button is not pressed, any value otherwise.

### 5.2.1.3 btn\_register\_fn()

```
void btn_register_fn (
    Button * b,
    void * function )
```

Registers a function to be called whether button is fired.

#### Parameters

<i>b</i>	Button instance.
<i>function</i>	Function pointer.

### 5.2.1.4 btn\_set\_mode()

```
void btn_set_mode (
    Button * b,
    Button_Mode mode )
```

Changes button behavior, waiting or not to be released to fire action.

#### Parameters

<i>b</i>	Button instance.
<i>mode</i>	New button mode.

### 5.2.1.5 btn\_setup()

```
Button* btn_setup (
    uint8_t pin_number )
```

Setups a new instance of a [Button](#).

#### Parameters

<i>number</i>	Digital pin number associated to the <a href="#">Button</a> .
---------------	---

#### Returns

A pointer to [Button](#) structure.

## 5.3 src/delay.c File Reference

```
#include "delay.h"
```

### Functions

- void [delay\\_1us](#) ()  
*Delays for, approximately, 1 microsecond.*
- void [delay\\_1ms](#) ()  
*Delays for, approximately, 1 millisecond.*
- void [delay\\_us](#) (uint32\_t time)  
*Delays for, approximately, a specified time, in microseconds.*
- void [delay\\_ms](#) (uint32\_t time)  
*Delays for, approximately, a specified time, in milliseconds.*

### 5.3.1 Function Documentation

#### 5.3.1.1 delay\_1ms()

```
void delay_1ms (
    void )
```

Delays for, approximately, 1 millisecond.



#### 5.3.1.2 delay\_1us()

```
void delay_1us (
    void )
```

Delays for, approximately, 1 microsecond.

#### 5.3.1.3 delay\_ms()

```
void delay_ms (
    uint32_t )
```

Delays for, approximately, a specified time, in milliseconds.

##### Parameters

<i>time</i>	Time for delay (in milliseconds)
-------------	----------------------------------

#### 5.3.1.4 delay\_us()

```
void delay_us (
    uint32_t )
```

Delays for, approximately, a specified time, in microseconds.

##### Parameters

<i>time</i>	Time for delay (in microseconds)
-------------	----------------------------------

## 5.4 src/include/bcd.h File Reference

```
#include "pins.h"
#include <stdlib.h>
```

### Data Structures

- struct [BCD](#)

## Functions

- `BCD * bcd_setup` (`uint8_t a`, `uint8_t b`, `uint8_t c`, `uint8_t d`, `uint8_t e`, `uint8_t f`, `uint8_t g`, `uint8_t dp`, `double delay`)  
*Setups a new instance of a display.*
- `void bcd_show_char` (`BCD *display`, `char character`)  
*Displays an ASCII character in a display.*
- `void bcd_show_string` (`BCD *display`, `char *str`)  
*Displays a string in a display, a character at a time, interleaved by a delay.*
- `void bcd_delay` (`BCD *display`)  
*Freezes a display for a presetted delay.*
- `void bcd_clear` (`BCD *display`)  
*Clears a display.*
- `void bcd_set_loop` (`BCD *display`, `uint8_t loop`)  
*Sets a loop mode for a display.*
- `void bcd_free` (`BCD *display`)  
*Destroys a display instance.*

### 5.4.1 Function Documentation

#### 5.4.1.1 `bcd_clear()`

```
void bcd_clear (
    BCD * display )
```

Clears a display.

##### Parameters

<code>display</code>	Display instance.
----------------------	-------------------

#### 5.4.1.2 `bcd_delay()`

```
void bcd_delay (
    BCD * display )
```

Freezes a display for a presetted delay.

##### Parameters

<code>display</code>	Display instance.
----------------------	-------------------

#### 5.4.1.3 bcd\_free()

```
void bcd_free (
    BCD * display )
```

Destroys a display instance.

##### Parameters

<i>display</i>	Display instance.
----------------	-------------------

#### 5.4.1.4 bcd\_set\_loop()

```
void bcd_set_loop (
    BCD * display,
    uint8_t loop )
```

Sets a loop mode for a display.

##### Parameters

<i>display</i>	Display instance.
<i>loop</i>	Defines a unique exhibition (0) or a cyclic exhibition (any other value).

#### 5.4.1.5 bcd\_setup()

```
BCD* bcd_setup (
    uint8_t a,
    uint8_t b,
    uint8_t c,
    uint8_t d,
    uint8_t e,
    uint8_t f,
    uint8_t g,
    uint8_t dp,
    double delay )
```

Setups a new instance of a display.

##### Parameters

<i>a</i>	Digital pin where the display pin 'a' is plugged.
<i>b</i>	Digital pin where the display pin 'b' is plugged.
<i>c</i>	Digital pin where the display pin 'c' is plugged.
<i>d</i>	Digital pin where the display pin 'd' is plugged.
<i>e</i>	Digital pin where the display pin 'e' is plugged.

**Parameters**

<i>f</i>	Digital pin where the display pin 'f' is plugged.
<i>g</i>	Digital pin where the display pin 'g' is plugged.
<i>dp</i>	Digital pin where the dot pin of display is plugged.
<i>delay</i>	Delay between transactions, if multiple characters are going to be shown.

**Returns**

A pointer to display structure.

**5.4.1.6 bcd\_show\_char()**

```
void bcd_show_char (
    BCD * display,
    char character )
```

Displays an ASCII character in a display.

**Parameters**

<i>display</i>	Display instance.
<i>character</i>	Character to be shown.

**5.4.1.7 bcd\_show\_string()**

```
void bcd_show_string (
    BCD * display,
    char * str )
```

Displays a string in a display, a character at a time, interleaved by a delay.

**Parameters**

<i>display</i>	Display instance.
<i>str</i>	String to be shown.

**5.5 src/include/bcd\_seven.h File Reference****Macros**

- #define `rep_a` 0

- Represents the display pin 'a'.*
- #define `rep_b` 1
- Represents the display pin 'b'.*
- #define `rep_c` 2
- Represents the display pin 'c'.*
- #define `rep_d` 3
- Represents the display pin 'd'.*
- #define `rep_e` 4
- Represents the display pin 'e'.*
- #define `rep_f` 5
- Represents the display pin 'f'.*
- #define `rep_g` 6
- Represents the display pin 'g'.*
- #define `rep_dp` 7
- Represents the display pin 'dp'.*

## Variables

- `uint8_t sseg_0 []` = {6, `rep_a`, `rep_b`, `rep_c`, `rep_d`, `rep_e`, `rep_f`}
- Specifies pins to write '0'.*
- `uint8_t sseg_1 []` = {2, `rep_b`, `rep_c`}
- Specifies pins to write '1'.*
- `uint8_t sseg_2 []` = {5, `rep_a`, `rep_b`, `rep_d`, `rep_e`, `rep_g`}
- Specifies pins to write '2'.*
- `uint8_t sseg_3 []` = {5, `rep_a`, `rep_b`, `rep_c`, `rep_d`, `rep_g`}
- Specifies pins to write '3'.*
- `uint8_t sseg_4 []` = {4, `rep_b`, `rep_c`, `rep_f`, `rep_g`}
- Specifies pins to write '4'.*
- `uint8_t sseg_5 []` = {5, `rep_a`, `rep_c`, `rep_d`, `rep_f`, `rep_g`}
- Specifies pins to write '5'.*
- `uint8_t sseg_6 []` = {6, `rep_a`, `rep_c`, `rep_d`, `rep_e`, `rep_f`, `rep_g`}
- Specifies pins to write '6'.*
- `uint8_t sseg_7 []` = {3, `rep_a`, `rep_b`, `rep_c`}
- Specifies pins to write '7'.*
- `uint8_t sseg_8 []` = {7, `rep_a`, `rep_b`, `rep_c`, `rep_d`, `rep_e`, `rep_f`, `rep_g`}
- Specifies pins to write '8'.*
- `uint8_t sseg_9 []` = {6, `rep_a`, `rep_b`, `rep_c`, `rep_d`, `rep_f`, `rep_g`}
- Specifies pins to write '9'.*
- `uint8_t sseg_A []` = {6, `rep_a`, `rep_b`, `rep_c`, `rep_e`, `rep_f`, `rep_g`}
- Specifies pins to write 'A'.*
- `uint8_t sseg_B []` = {5, `rep_c`, `rep_d`, `rep_e`, `rep_f`, `rep_g`}
- Specifies pins to write 'B'.*
- `uint8_t sseg_C []` = {4, `rep_a`, `rep_d`, `rep_e`, `rep_f`}
- Specifies pins to write 'C'.*
- `uint8_t sseg_D []` = {5, `rep_b`, `rep_c`, `rep_d`, `rep_e`, `rep_g`}
- Specifies pins to write 'D'.*
- `uint8_t sseg_E []` = {5, `rep_a`, `rep_d`, `rep_e`, `rep_f`, `rep_g`}
- Specifies pins to write 'E'.*
- `uint8_t sseg_F []` = {4, `rep_a`, `rep_e`, `rep_f`, `rep_g`}
- Specifies pins to write 'F'.*

- `uint8_t sseg_G [] = {5, rep_a, rep_c, rep_d, rep_e, rep_f}`  
*Specifies pins to write 'G'.*
- `uint8_t sseg_H [] = {5, rep_b, rep_c, rep_e, rep_f, rep_g}`  
*Specifies pins to write 'H'.*
- `uint8_t sseg_I [] = {2, rep_e, rep_f}`  
*Specifies pins to write 'I'.*
- `uint8_t sseg_J [] = {4, rep_b, rep_c, rep_d, rep_e}`  
*Specifies pins to write 'J'.*
- `uint8_t sseg_K [] = {5, rep_a, rep_c, rep_e, rep_f, rep_g}`  
*Specifies pins to write 'K'.*
- `uint8_t sseg_L [] = {3, rep_d, rep_e, rep_f}`  
*Specifies pins to write 'L'.*
- `uint8_t sseg_M [] = {3, rep_a, rep_c, rep_e}`  
*Specifies pins to write 'M'.*
- `uint8_t sseg_N [] = {5, rep_a, rep_b, rep_c, rep_e, rep_f}`  
*Specifies pins to write 'N'.*
- `uint8_t sseg_O [] = {6, rep_a, rep_b, rep_c, rep_d, rep_e, rep_f}`  
*Specifies pins to write 'O'.*
- `uint8_t sseg_P [] = {5, rep_a, rep_b, rep_e, rep_f, rep_g}`  
*Specifies pins to write 'P'.*
- `uint8_t sseg_Q [] = {5, rep_a, rep_b, rep_d, rep_f, rep_g}`  
*Specifies pins to write 'Q'.*
- `uint8_t sseg_R [] = {4, rep_a, rep_b, rep_e, rep_f}`  
*Specifies pins to write 'R'.*
- `uint8_t sseg_S [] = {5, rep_a, rep_c, rep_d, rep_f, rep_g}`  
*Specifies pins to write 'S'.*
- `uint8_t sseg_T [] = {4, rep_d, rep_e, rep_f, rep_g}`  
*Specifies pins to write 'T'.*
- `uint8_t sseg_U [] = {5, rep_b, rep_c, rep_d, rep_e, rep_f}`  
*Specifies pins to write 'U'.*
- `uint8_t sseg_V [] = {5, rep_b, rep_c, rep_d, rep_e, rep_f}`  
*Specifies pins to write 'V'.*
- `uint8_t sseg_W [] = {3, rep_b, rep_d, rep_f}`  
*Specifies pins to write 'W'.*
- `uint8_t sseg_X [] = {5, rep_b, rep_c, rep_e, rep_f, rep_g}`  
*Specifies pins to write 'X'.*
- `uint8_t sseg_Y [] = {5, rep_b, rep_c, rep_d, rep_f, rep_g}`  
*Specifies pins to write 'Y'.*
- `uint8_t sseg_Z [] = {5, rep_a, rep_b, rep_d, rep_e, rep_g}`  
*Specifies pins to write 'Z'.*
- `uint8_t sseg_a [] = {6, rep_a, rep_b, rep_c, rep_d, rep_e, rep_g}`  
*Specifies pins to write 'a'.*
- `uint8_t sseg_b [] = {5, rep_c, rep_d, rep_e, rep_f, rep_g}`  
*Specifies pins to write 'b'.*
- `uint8_t sseg_c [] = {3, rep_d, rep_e, rep_g}`  
*Specifies pins to write 'c'.*
- `uint8_t sseg_d [] = {5, rep_b, rep_c, rep_d, rep_e, rep_g}`  
*Specifies pins to write 'd'.*
- `uint8_t sseg_e [] = {6, rep_a, rep_b, rep_d, rep_e, rep_f, rep_g}`  
*Specifies pins to write 'e'.*
- `uint8_t sseg_f [] = {4, rep_a, rep_e, rep_f, rep_g}`

- Specifies pins to write 'f'.*

  - `uint8_t sseg_g [] = {6, rep_a, rep_b, rep_c, rep_d, rep_f, rep_g}`
- Specifies pins to write 'g'.*

  - `uint8_t sseg_h [] = {4, rep_c, rep_e, rep_f, rep_g}`
- Specifies pins to write 'h'.*

  - `uint8_t sseg_i [] = {1, rep_e}`
- Specifies pins to write 'i'.*

  - `uint8_t sseg_j [] = {2, rep_c, rep_d}`
- Specifies pins to write 'j'.*

  - `uint8_t sseg_k [] = {5, rep_a, rep_c, rep_e, rep_f, rep_g}`
- Specifies pins to write 'k'.*

  - `uint8_t sseg_l [] = {2, rep_e, rep_f}`
- Specifies pins to write 'l'.*

  - `uint8_t sseg_m [] = {2, rep_c, rep_e}`
- Specifies pins to write 'm'.*

  - `uint8_t sseg_n [] = {3, rep_c, rep_e, rep_g}`
- Specifies pins to write 'n'.*

  - `uint8_t sseg_o [] = {4, rep_c, rep_d, rep_e, rep_g}`
- Specifies pins to write 'o'.*

  - `uint8_t sseg_p [] = {5, rep_a, rep_b, rep_e, rep_f, rep_g}`
- Specifies pins to write 'p'.*

  - `uint8_t sseg_q [] = {5, rep_a, rep_b, rep_c, rep_f, rep_g}`
- Specifies pins to write 'q'.*

  - `uint8_t sseg_r [] = {2, rep_e, rep_g}`
- Specifies pins to write 'r'.*

  - `uint8_t sseg_s [] = {5, rep_a, rep_c, rep_d, rep_f, rep_g}`
- Specifies pins to write 's'.*

  - `uint8_t sseg_t [] = {4, rep_d, rep_e, rep_f, rep_g}`
- Specifies pins to write 't'.*

  - `uint8_t sseg_u [] = {3, rep_c, rep_d, rep_e}`
- Specifies pins to write 'u'.*

  - `uint8_t sseg_v [] = {3, rep_c, rep_d, rep_e}`
- Specifies pins to write 'v'.*

  - `uint8_t sseg_w [] = {2, rep_c, rep_e}`
- Specifies pins to write 'w'.*

  - `uint8_t sseg_x [] = {5, rep_b, rep_c, rep_e, rep_f, rep_g}`
- Specifies pins to write 'x'.*

  - `uint8_t sseg_y [] = {5, rep_b, rep_c, rep_d, rep_f, rep_g}`
- Specifies pins to write 'y'.*

  - `uint8_t sseg_z [] = {5, rep_a, rep_b, rep_d, rep_e, rep_g}`
- Specifies pins to write 'z'.*

  - `const uint8_t * SevenSegmentASCII []`

*7-Seg vector that maps which pins must be turned on to draw each supported ASCII character.*

### 5.5.1 Macro Definition Documentation

#### 5.5.1.1 rep\_a

```
#define rep_a 0
```

Represents the display pin 'a'.

#### 5.5.1.2 rep\_b

```
#define rep_b 1
```

Represents the display pin 'b'.

#### 5.5.1.3 rep\_c

```
#define rep_c 2
```

Represents the display pin 'c'.

#### 5.5.1.4 rep\_d

```
#define rep_d 3
```

Represents the display pin 'd'.

#### 5.5.1.5 rep\_dp

```
#define rep_dp 7
```

Represents the display pin 'dp'.

#### 5.5.1.6 rep\_e

```
#define rep_e 4
```

Represents the display pin 'e'.



### 5.5.1.7 rep\_f

```
#define rep_f 5
```

Represents the display pin 'f'.

### 5.5.1.8 rep\_g

```
#define rep_g 6
```

Represents the display pin 'g'.

## 5.5.2 Variable Documentation

### 5.5.2.1 SevenSegmentASCII

```
const uint8_t* SevenSegmentASCII[]
```

**Initial value:**

```
= {
    sseg_0, sseg_1, sseg_2, sseg_3, sseg_4, sseg_5,      sseg_6, sseg_7, sseg_8, sseg_9, sseg_A, sseg_B,
        sseg_C, sseg_D, sseg_E, sseg_F, sseg_G, sseg_H,
        sseg_I, sseg_J, sseg_K, sseg_L, sseg_M, sseg_N,
        sseg_O, sseg_P, sseg_Q, sseg_R, sseg_S, sseg_T,
        sseg_U, sseg_V, sseg_W, sseg_X, sseg_Y, sseg_Z,
        sseg_a, sseg_b, sseg_c, sseg_d, sseg_e, sseg_f,
        sseg_g, sseg_h, sseg_i, sseg_j, sseg_k, sseg_l,
        sseg_m, sseg_n, sseg_o, sseg_p, sseg_q, sseg_r,
        sseg_s, sseg_t, sseg_u, sseg_v, sseg_w, sseg_x,      sseg_y, sseg_z
}
```

7-Seg vector that maps which pins must be turned on to draw each supported ASCII character.

### 5.5.2.2 sseg\_0

```
uint8_t sseg_0[] = {6, rep_a, rep_b, rep_c, rep_d, rep_e, rep_f}
```

Specifies pins to write '0'.

### 5.5.2.3 sseg\_1

```
uint8_t sseg_1[] = {2, rep_b, rep_c}
```

Specifies pins to write '1'.

#### 5.5.2.4 sseg\_2

```
uint8_t sseg_2[] = {5, rep_a, rep_b, rep_d, rep_e, rep_g}
```

Specifies pins to write '2'.

#### 5.5.2.5 sseg\_3

```
uint8_t sseg_3[] = {5, rep_a, rep_b, rep_c, rep_d, rep_g}
```

Specifies pins to write '3'.

#### 5.5.2.6 sseg\_4

```
uint8_t sseg_4[] = {4, rep_b, rep_c, rep_f, rep_g}
```

Specifies pins to write '4'.

#### 5.5.2.7 sseg\_5

```
uint8_t sseg_5[] = {5, rep_a, rep_c, rep_d, rep_f, rep_g}
```

Specifies pins to write '5'.

#### 5.5.2.8 sseg\_6

```
uint8_t sseg_6[] = {6, rep_a, rep_c, rep_d, rep_e, rep_f, rep_g}
```

Specifies pins to write '6'.

#### 5.5.2.9 sseg\_7

```
uint8_t sseg_7[] = {3, rep_a, rep_b, rep_c}
```

Specifies pins to write '7'.

#### 5.5.2.10 sseg\_8

```
uint8_t sseg_8[] = {7, rep_a, rep_b, rep_c, rep_d, rep_e, rep_f, rep_g}
```

Specifies pins to write '8'.

#### 5.5.2.11 sseg\_9

```
uint8_t sseg_9[] = {6, rep_a, rep_b, rep_c, rep_d, rep_f, rep_g}
```

Specifies pins to write '9'.

#### 5.5.2.12 sseg\_A

```
uint8_t sseg_A[] = {6, rep_a, rep_b, rep_c, rep_e, rep_f, rep_g}
```

Specifies pins to write 'A'.

#### 5.5.2.13 sseg\_a

```
uint8_t sseg_a[] = {6, rep_a, rep_b, rep_c, rep_d, rep_e, rep_g}
```

Specifies pins to write 'a'.

#### 5.5.2.14 sseg\_B

```
uint8_t sseg_B[] = {5, rep_c, rep_d, rep_e, rep_f, rep_g}
```

Specifies pins to write 'B'.

#### 5.5.2.15 sseg\_b

```
uint8_t sseg_b[] = {5, rep_c, rep_d, rep_e, rep_f, rep_g}
```

Specifies pins to write 'b'.

#### 5.5.2.16 sseg\_C

```
uint8_t sseg_C[] = {4, rep_a, rep_d, rep_e, rep_f}
```

Specifies pins to write 'C'.

#### 5.5.2.17 sseg\_c

```
uint8_t sseg_c[] = {3, rep_d, rep_e, rep_g}
```

Specifies pins to write 'c'.

#### 5.5.2.18 sseg\_D

```
uint8_t sseg_D[] = {5, rep_b, rep_c, rep_d, rep_e, rep_g}
```

Specifies pins to write 'D'.

#### 5.5.2.19 sseg\_d

```
uint8_t sseg_d[] = {5, rep_b, rep_c, rep_d, rep_e, rep_g}
```

Specifies pins to write 'd'.

#### 5.5.2.20 sseg\_E

```
uint8_t sseg_E[] = {5, rep_a, rep_d, rep_e, rep_f, rep_g}
```

Specifies pins to write 'E'.

#### 5.5.2.21 sseg\_e

```
uint8_t sseg_e[] = {6, rep_a, rep_b, rep_d, rep_e, rep_f, rep_g}
```

Specifies pins to write 'e'.

#### 5.5.2.22 sseg\_F

```
uint8_t sseg_F[] = {4, rep_a, rep_e, rep_f, rep_g}
```

Specifies pins to write 'F'.

#### 5.5.2.23 sseg\_f

```
uint8_t sseg_f[] = {4, rep_a, rep_e, rep_f, rep_g}
```

Specifies pins to write 'f'.

#### 5.5.2.24 sseg\_G

```
uint8_t sseg_G[] = {5, rep_a, rep_c, rep_d, rep_e, rep_f}
```

Specifies pins to write 'G'.

#### 5.5.2.25 sseg\_g

```
uint8_t sseg_g[] = {6, rep_a, rep_b, rep_c, rep_d, rep_f, rep_g}
```

Specifies pins to write 'g'.

#### 5.5.2.26 sseg\_H

```
uint8_t sseg_H[] = {5, rep_b, rep_c, rep_e, rep_f, rep_g}
```

Specifies pins to write 'H'.

#### 5.5.2.27 sseg\_h

```
uint8_t sseg_h[] = {4, rep_c, rep_e, rep_f, rep_g}
```

Specifies pins to write 'h'.

#### 5.5.2.28 sseg\_I

```
uint8_t sseg_I[] = {2, rep_e, rep_f}
```

Specifies pins to write 'I'.

#### 5.5.2.29 sseg\_i

```
uint8_t sseg_i[] = {1, rep_e}
```

Specifies pins to write 'i'.

#### 5.5.2.30 sseg\_J

```
uint8_t sseg_J[] = {4, rep_b, rep_c, rep_d, rep_e}
```

Specifies pins to write 'J'.

#### 5.5.2.31 sseg\_j

```
uint8_t sseg_j[] = {2, rep_c, rep_d}
```

Specifies pins to write 'j'.

#### 5.5.2.32 sseg\_K

```
uint8_t sseg_K[] = {5, rep_a, rep_c, rep_e, rep_f, rep_g}
```

Specifies pins to write 'K'.

#### 5.5.2.33 sseg\_k

```
uint8_t sseg_k[] = {5, rep_a, rep_c, rep_e, rep_f, rep_g}
```

Specifies pins to write 'k'.

#### 5.5.2.34 sseg\_L

```
uint8_t sseg_L[] = {3, rep_d, rep_e, rep_f}
```

Specifies pins to write 'L'.

#### 5.5.2.35 sseg\_I

```
uint8_t sseg_I[] = {2, rep_e, rep_f}
```

Specifies pins to write 'I'.

#### 5.5.2.36 sseg\_M

```
uint8_t sseg_M[] = {3, rep_a, rep_c, rep_e}
```

Specifies pins to write 'M'.

#### 5.5.2.37 sseg\_m

```
uint8_t sseg_m[] = {2, rep_c, rep_e}
```

Specifies pins to write 'm'.

#### 5.5.2.38 sseg\_N

```
uint8_t sseg_N[] = {5, rep_a, rep_b, rep_c, rep_e, rep_f}
```

Specifies pins to write 'N'.

#### 5.5.2.39 sseg\_n

```
uint8_t sseg_n[] = {3, rep_c, rep_e, rep_g}
```

Specifies pins to write 'n'.

#### 5.5.2.40 sseg\_O

```
uint8_t sseg_O[] = {6, rep_a, rep_b, rep_c, rep_d, rep_e, rep_f}
```

Specifies pins to write 'O'.

#### 5.5.2.41 sseg\_o

```
uint8_t sseg_o[] = {4, rep_c, rep_d, rep_e, rep_g}
```

Specifies pins to write 'o'.

#### 5.5.2.42 sseg\_P

```
uint8_t sseg_P[] = {5, rep_a, rep_b, rep_e, rep_f, rep_g}
```

Specifies pins to write 'P'.

#### 5.5.2.43 sseg\_p

```
uint8_t sseg_p[] = {5, rep_a, rep_b, rep_e, rep_f, rep_g}
```

Specifies pins to write 'p'.

#### 5.5.2.44 sseg\_Q

```
uint8_t sseg_Q[] = {5, rep_a, rep_b, rep_d, rep_f, rep_g}
```

Specifies pins to write 'Q'.

#### 5.5.2.45 sseg\_q

```
uint8_t sseg_q[] = {5, rep_a, rep_b, rep_c, rep_f, rep_g}
```

Specifies pins to write 'q'.



#### 5.5.2.46 sseg\_R

```
uint8_t sseg_R[] = {4, rep_a, rep_b, rep_e, rep_f}
```

Specifies pins to write 'R'.

#### 5.5.2.47 sseg\_r

```
uint8_t sseg_r[] = {2, rep_e, rep_g}
```

Specifies pins to write 'r'.

#### 5.5.2.48 sseg\_S

```
uint8_t sseg_S[] = {5, rep_a, rep_c, rep_d, rep_f, rep_g}
```

Specifies pins to write 'S'.

#### 5.5.2.49 sseg\_s

```
uint8_t sseg_s[] = {5, rep_a, rep_c, rep_d, rep_f, rep_g}
```

Specifies pins to write 's'.

#### 5.5.2.50 sseg\_T

```
uint8_t sseg_T[] = {4, rep_d, rep_e, rep_f, rep_g}
```

Specifies pins to write 'T'.

#### 5.5.2.51 sseg\_t

```
uint8_t sseg_t[] = {4, rep_d, rep_e, rep_f, rep_g}
```

Specifies pins to write 't'.

#### 5.5.2.52 sseg\_U

```
uint8_t sseg_U[] = {5, rep_b, rep_c, rep_d, rep_e, rep_f}
```

Specifies pins to write 'U'.

#### 5.5.2.53 sseg\_u

```
uint8_t sseg_u[] = {3, rep_c, rep_d, rep_e}
```

Specifies pins to write 'u'.

#### 5.5.2.54 sseg\_V

```
uint8_t sseg_V[] = {5, rep_b, rep_c, rep_d, rep_e, rep_f}
```

Specifies pins to write 'V'.

#### 5.5.2.55 sseg\_v

```
uint8_t sseg_v[] = {3, rep_c, rep_d, rep_e}
```

Specifies pins to write 'v'.

#### 5.5.2.56 sseg\_W

```
uint8_t sseg_W[] = {3, rep_b, rep_d, rep_f}
```

Specifies pins to write 'W'.

#### 5.5.2.57 sseg\_w

```
uint8_t sseg_w[] = {2, rep_c, rep_e}
```

Specifies pins to write 'w'.

#### 5.5.2.58 sseg\_X

```
uint8_t sseg_X[] = {5, rep_b, rep_c, rep_e, rep_f, rep_g}
```

Specifies pins to write 'X'.

#### 5.5.2.59 sseg\_x

```
uint8_t sseg_x[] = {5, rep_b, rep_c, rep_e, rep_f, rep_g}
```

Specifies pins to write 'x'.

#### 5.5.2.60 sseg\_Y

```
uint8_t sseg_Y[] = {5, rep_b, rep_c, rep_d, rep_f, rep_g}
```

Specifies pins to write 'Y'.

#### 5.5.2.61 sseg\_y

```
uint8_t sseg_y[] = {5, rep_b, rep_c, rep_d, rep_f, rep_g}
```

Specifies pins to write 'y'.

#### 5.5.2.62 sseg\_Z

```
uint8_t sseg_Z[] = {5, rep_a, rep_b, rep_d, rep_e, rep_g}
```

Specifies pins to write 'Z'.

#### 5.5.2.63 sseg\_z

```
uint8_t sseg_z[] = {5, rep_a, rep_b, rep_d, rep_e, rep_g}
```

Specifies pins to write 'z'.

## 5.6 src/include/button.h File Reference

```
#include "pins.h"
```

### Data Structures

- struct [Button](#)

### Typedefs

- typedef enum [button\\_mode](#) [Button\\_Mode](#)  
*[Button](#) mode options.*

### Enumerations

- enum [button\\_mode](#) { [Btn\\_Up](#) = 0, [Btn\\_Down](#) = 1 }

### Functions

- [Button](#) \* [btn\\_setup](#) (uint8\_t pin\_number)  
*Setups a new instance of a [Button](#).*
- void [btn\\_register\\_fn](#) ([Button](#) \*b, void \*function)  
*Registers a function to be called whether button is fired.*
- void [btn\\_set\\_mode](#) ([Button](#) \*b, [Button\\_Mode](#) mode)  
*Changes button behavior, waiting or not to be released to fire action.*
- void [btn\\_fired](#) ([Button](#) \*b)  
*Verifies if a button is pressed at moment and fires it's action.*
- uint8\_t [btn\\_pressed](#) ([Button](#) \*b)  
*Verifies if a button is pressed at moment.*

### 5.6.1 Typedef Documentation

#### 5.6.1.1 [Button\\_Mode](#)

```
typedef enum button\_mode Button\_Mode
```

[Button](#) mode options.

### 5.6.2 Enumeration Type Documentation

#### 5.6.2.1 [button\\_mode](#)

```
enum button\_mode
```

## Enumerator

Btn_Up	Fires action immediately after button get pressed.
Btn_Down	Fires action when button is released.

### 5.6.3 Function Documentation

#### 5.6.3.1 btn\_fired()

```
void btn_fired (
    Button * b )
```

Verifies if a button is pressed at moment and fires it's action.

## Parameters

<i>b</i>	Button instance.
----------	------------------

#### 5.6.3.2 btn\_pressed()

```
uint8_t btn_pressed (
    Button * b )
```

Verifies if a button is pressed at moment.

## Parameters

<i>b</i>	Button instance.
----------	------------------

## Returns

Returns 0 if button is not pressed, any value otherwise.

#### 5.6.3.3 btn\_register\_fn()

```
void btn_register_fn (
    Button * b,
    void * function )
```

Registers a function to be called whether button is fired.

## Parameters

<i>b</i>	<a href="#">Button</a> instance.
<i>function</i>	Function pointer.

## 5.6.3.4 btn\_set\_mode()

```
void btn_set_mode (
    Button * b,
    Button\_Mode mode )
```

Changes button behavior, waiting or not to be released to fire action.

## Parameters

<i>b</i>	<a href="#">Button</a> instance.
<i>mode</i>	New button mode.

## 5.6.3.5 btn\_setup()

```
Button* btn_setup (
    uint8_t pin_number )
```

Setups a new instance of a [Button](#).

## Parameters

<i>number</i>	Digital pin number associated to the <a href="#">Button</a> .
---------------	---

## Returns

A pointer to [Button](#) structure.

## 5.7 src/include/delay.h File Reference

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

## Functions

- void [delay\\_1us](#) (void)

- Delays for, approximately, 1 microsecond.*
  - void `delay_1ms` (void)  
*Delays for, approximately, 1 millisecond.*
- void `delay_us` (uint32\_t)  
*Delays for, approximately, a specified time, in microseconds.*
- void `delay_ms` (uint32\_t)  
*Delays for, approximately, a specified time, in milliseconds.*

## 5.7.1 Function Documentation

### 5.7.1.1 `delay_1ms()`

```
void delay_1ms (  
    void )
```

Delays for, approximately, 1 millisecond.

### 5.7.1.2 `delay_us()`

```
void delay_us (  
    void )
```

Delays for, approximately, 1 microsecond.

### 5.7.1.3 `delay_ms()`

```
void delay_ms (  
    uint32_t )
```

Delays for, approximately, a specified time, in milliseconds.

#### Parameters

<i>time</i>	Time for delay (in milliseconds)
-------------	----------------------------------

### 5.7.1.4 `delay_us()`

```
void delay_us (  
    uint32_t )
```

Delays for, approximately, a specified time, in microseconds.

#### Parameters

<i>time</i>	Time for delay (in microseconds)
-------------	----------------------------------

## 5.8 src/include/led.h File Reference

```
#include "pins.h"
```

### Data Structures

- struct [Led](#)

### Typedefs

- typedef enum [led\\_state](#) [Led\\_State](#)  
*Maps a possible [Led](#) state.*

### Enumerations

- enum [led\\_state](#) { [On](#) = 1, [Off](#) = 0 }

### Functions

- [Led](#) \* [led\\_setup](#) (int number)  
*Setups a new instance of a [Led](#).*
- void [led\\_make\\_light](#) ([Led](#) \*led)  
*Turns on the instance [Led](#)'s light.*
- void [led\\_kill\\_light](#) ([Led](#) \*led)  
*Turns off the instance [Led](#)'s light.*
- void [led\\_switch\\_light](#) ([Led](#) \*led)  
*Toggles the instance [Led](#)'s light.*
- void [led\\_blink](#) ([Led](#) \*led, double time\_ms)  
*Blinks the instance [Led](#)'s light, keeping it on and off for a specified time.*
- void [led\\_free](#) ([Led](#) \*led)  
*Destroys a [Led](#) instance.*

#### 5.8.1 Typedef Documentation



### 5.8.1.1 Led\_State

```
typedef enum led_state Led_State
```

Maps a possible [Led](#) state.

## 5.8.2 Enumeration Type Documentation

### 5.8.2.1 led\_state

```
enum led_state
```

#### Enumerator

On	<a href="#">Led</a> 's light is on.
Off	<a href="#">Led</a> 's light is off.

## 5.8.3 Function Documentation

### 5.8.3.1 led\_blink()

```
void led_blink (  
    Led * led,  
    double time_ms )
```

Blinks the instance [Led](#)'s light, keeping it on and off for a specified time.

#### Parameters

<i>led</i>	<a href="#">Led</a> instance.
<i>time_ms</i>	Time interval to keep <a href="#">Led</a> 's light on and off.

### 5.8.3.2 led\_free()

```
void led_free (  
    Led * led )
```

Destroys a [Led](#) instance.

**Parameters**

<i>led</i>	<a href="#">Led</a> instance.
------------	-------------------------------

**5.8.3.3 led\_kill\_light()**

```
void led_kill_light (
    Led * led )
```

Turns off the instance [Led](#)'s light.

**Parameters**

<i>led</i>	<a href="#">Led</a> instance.
------------	-------------------------------

**5.8.3.4 led\_make\_light()**

```
void led_make_light (
    Led * led )
```

Turns on the instance [Led](#)'s light.

**Parameters**

<i>led</i>	<a href="#">Led</a> instance.
------------	-------------------------------

**5.8.3.5 led\_setup()**

```
Led* led_setup (
    int number )
```

Setups a new instance of a [Led](#).

**Parameters**

<i>number</i>	Digital pin number associated to the <a href="#">Led</a> .
---------------	--

**Returns**

A pointer to [Led](#) structure.

## 5.8.3.6 led\_switch\_light()

```
void led_switch_light (
    Led * led )
```

Toggles the instance [Led](#)'s light.

## Parameters

<i>led</i>	<a href="#">Led</a> instance.
------------	-------------------------------

## 5.9 src/include/macros.h File Reference

## Macros

- #define [MACROS](#)
- #define [swt\\_bit](#)(x, n) (x ^= (1 << n))
- #define [set\\_bit](#)(x, n) (x |= (1 << n))
- #define [clr\\_bit](#)(x, n) (x &= ~(1 << n))
- #define [tst\\_bit](#)(x, n) (x & (1 << n))
- #define [comp\\_masks](#)(x, y) ((1 << x) | (1 << y))

## 5.9.1 Macro Definition Documentation

## 5.9.1.1 clr\_bit

```
#define clr_bit(
    x,
    n ) (x &= ~(1 << n))
```

## 5.9.1.2 comp\_masks

```
#define comp_masks(
    x,
    y ) ((1 << x) | (1 << y))
```

## 5.9.1.3 MACROS

```
#define MACROS
```

#### 5.9.1.4 set\_bit

```
#define set_bit(  
    x,  
    n ) (x |= (1 << n))
```

#### 5.9.1.5 swt\_bit

```
#define swt_bit(  
    x,  
    n ) (x ^= (1 << n))
```

#### 5.9.1.6 tst\_bit

```
#define tst_bit(  
    x,  
    n ) (x & (1 << n))
```

### 5.10 src/include/pins.h File Reference

```
#include <avr/io.h>  
#include <avr/portpins.h>  
#include <util/delay.h>  
#include "macros.h"  
#include <stdlib.h>
```

#### Data Structures

- struct [Pin](#)
- struct [bytes](#)

#### Macros

- #define [F\\_CPU](#) 16000000UL
- #define [HIGH](#) 1
- #define [LOW](#) 0

## Functions

- uint8\_t [set\\_output](#) (uint8\_t number)  
*Sets a pin as a output, using its DDR.*
- uint8\_t [set\\_pullup](#) (uint8\_t number)  
*Sets a pin as active, using its PORT.*
- uint8\_t [toggle\\_pullup](#) (uint8\_t number)  
*Toggles a pin activity, using its PORT.*
- uint8\_t [test\\_input](#) (uint8\_t number)  
*Tests if there is an input signal on a pin, using its PIN.*
- uint8\_t [set\\_input](#) (uint8\_t number)  
*Sets a pin as an input, using its DDR.*
- uint8\_t [clr\\_pullup](#) (uint8\_t number)  
*Sets a pin as inactive, using its PORT.*
- uint8\_t [digital\\_write](#) (uint8\_t number, int value)  
*Writes an integer value to a pin, using its PORT.*

## Variables

- [Pin Pins](#) [14]  
*Digital pin instances, common to all files.*
- [bytes Bytes](#) [14]  
*Bytes instances.*

## 5.10.1 Macro Definition Documentation

### 5.10.1.1 F\_CPU

```
#define F_CPU 16000000UL
```

### 5.10.1.2 HIGH

```
#define HIGH 1
```

### 5.10.1.3 LOW

```
#define LOW 0
```

## 5.10.2 Function Documentation

### 5.10.2.1 clr\_pullup()

```
uint8_t clr_pullup (
    uint8_t number )
```

Sets a pin as inactive, using its PORT.

**Parameters**

<i>number</i>	Digital pin number.
---------------	---------------------

**5.10.2.2 digital\_write()**

```
uint8_t digital_write (
    uint8_t number,
    int value )
```

Writes an integer value to a pin, using its PORT.

**Parameters**

<i>number</i>	Digital pin number.
---------------	---------------------

**5.10.2.3 set\_input()**

```
uint8_t set_input (
    uint8_t number )
```

Sets a pin as an input, using its DDR.

**Parameters**

<i>number</i>	Digital pin number.
---------------	---------------------

**5.10.2.4 set\_output()**

```
uint8_t set_output (
    uint8_t number )
```

Sets a pin as a output, using its DDR.

**Parameters**

<i>number</i>	Digital pin number.
---------------	---------------------

#### 5.10.2.5 set\_pullup()

```
uint8_t set_pullup (
    uint8_t number )
```

Sets a pin as active, using its PORT.

##### Parameters

<i>number</i>	Digital pin number.
---------------	---------------------

#### 5.10.2.6 test\_input()

```
uint8_t test_input (
    uint8_t number )
```

Tests if there is an input signal on a pin, using its PIN.

##### Parameters

<i>number</i>	Digital pin number.
---------------	---------------------

#### 5.10.2.7 toggle\_pullup()

```
uint8_t toggle_pullup (
    uint8_t number )
```

Toggles a pin activity, using its PORT.

##### Parameters

<i>number</i>	Digital pin number.
---------------	---------------------

### 5.10.3 Variable Documentation

#### 5.10.3.1 Bytes

`bytes` Bytes[14]

Bytes instances.

### 5.10.3.2 Pins

```
Pin Pins[14]
```

Digital pin instances, common to all files.

## 5.11 src/include/pwm.h File Reference

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

### Functions

- float `lookup_prescalar` (float)  
*Specifies the prescalar based on frequency.*
- int `lookup_ocr` (float)  
*Specifies OCR value based on frequency.*
- uint8\_t `lookup_cs` (float)  
*Specifies the CS control bits value based on frequency.*
- void `sqrwave` (uint8\_t, float)  
*Generates square waves pulse in a pin based on frequency.*
- void `pwmwave` (uint8\_t, uint8\_t)  
*Generates PWM waves in a pin based on a duty cycle.*
- uint8\_t `lookup_cs2` (float freq)

### 5.11.1 Function Documentation

#### 5.11.1.1 `lookup_cs()`

```
uint8_t lookup_cs (
    float )
```

Specifies the CS control bits value based on frequency.

#### Parameters

<i>freq</i>	Frequency value (Hz)
-------------	----------------------

#### Returns

CS control bits value



#### 5.11.1.2 lookup\_cs2()

```
uint8_t lookup_cs2 (
    float freq )
```

#### 5.11.1.3 lookup\_ocr()

```
int lookup_ocr (
    float )
```

Specifies OCR value based on frequency.

##### Parameters

<i>freq</i>	Frequency value (Hz)
-------------	----------------------

##### Returns

OCR value

#### 5.11.1.4 lookup\_prescalar()

```
float lookup_prescalar (
    float )
```

Specifies the prescalar based on frequency.

##### Parameters

<i>freq</i>	Frequency value (Hz)
-------------	----------------------

##### Returns

Prescalar value

#### 5.11.1.5 pwmwave()

```
void pwmwave (
    uint8_t ,
    uint8_t )
```

Generates PWM waves in a pin based on a duty cycle.

**Parameters**

<i>pin</i>	Pin number
<i>dutycycle</i>	Duty cycle value (0 ~ 255)

**5.11.1.6 sqwave()**

```
void sqwave (
    uint8_t ,
    float )
```

Generates square waves pulse in a pin based on frequency.

**Parameters**

<i>pin</i>	Pin number
<i>freq</i>	Frequency value (Hz)

**5.12 src/include/supersonic.h File Reference**

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

**Functions**

- int [supersonic](#) (uint8\_t, uint8\_t)  
*Measures a distance based on a sonar.*

**5.12.1 Function Documentation****5.12.1.1 supersonic()**

```
int supersonic (
    uint8_t ,
    uint8_t )
```

Measures a distance based on a sonar.

#### Parameters

<i>trig</i>	Digital pin number associated to the Trigger pin.
<i>echo</i>	Digital pin number associated to the Echo pin.

#### Returns

Measured distance (cm) between obstacle.

## 5.13 src/include/uart.h File Reference

```
#include <stdio.h>
```

#### Macros

- `#define F_CPU 16000000UL`
- `#define BAUD 9600`

#### Functions

- void `uart_init` (void)
- int `uart_putchar` (char c, FILE \*stream)
- int `uart_getchar` (FILE \*stream)

### 5.13.1 Macro Definition Documentation

#### 5.13.1.1 BAUD

```
#define BAUD 9600
```

#### 5.13.1.2 F\_CPU

```
#define F_CPU 16000000UL
```

### 5.13.2 Function Documentation

#### 5.13.2.1 `uart_getchar()`

```
int uart_getchar (
    FILE * stream )
```

#### 5.13.2.2 `uart_init()`

```
void uart_init (
    void )
```

#### 5.13.2.3 `uart_putchar()`

```
int uart_putchar (
    char c,
    FILE * stream )
```

### 5.14 `src/led.c` File Reference

```
#include "led.h"
```

#### Functions

- `Led * led_setup` (int number)  
*Setups a new instance of a `Led`.*
- `void led_make_light` (`Led *led`)  
*Turns on the instance `Led`'s light.*
- `void led_kill_light` (`Led *led`)  
*Turns off the instance `Led`'s light.*
- `void led_switch_light` (`Led *led`)  
*Toggles the instance `Led`'s light.*
- `void led_blink` (`Led *led`, double time\_ms)  
*Blinks the instance `Led`'s light, keeping it on and off for a specified time.*
- `void led_free` (`Led *led`)  
*Destroys a `Led` instance.*

#### 5.14.1 Function Documentation

##### 5.14.1.1 `led_blink()`

```
void led_blink (
    Led * led,
    double time_ms )
```

Blinks the instance `Led`'s light, keeping it on and off for a specified time.

## Parameters

<i>led</i>	<a href="#">Led</a> instance.
<i>time_ms</i>	Time interval to keep <a href="#">Led</a> 's light on and off.

5.14.1.2 `led_free()`

```
void led_free (  
    Led * led )
```

Destroys a [Led](#) instance.

## Parameters

<i>led</i>	<a href="#">Led</a> instance.
------------	-------------------------------

5.14.1.3 `led_kill_light()`

```
void led_kill_light (  
    Led * led )
```

Turns off the instance [Led](#)'s light.

## Parameters

<i>led</i>	<a href="#">Led</a> instance.
------------	-------------------------------

5.14.1.4 `led_make_light()`

```
void led_make_light (  
    Led * led )
```

Turns on the instance [Led](#)'s light.

## Parameters

<i>led</i>	<a href="#">Led</a> instance.
------------	-------------------------------

#### 5.14.1.5 led\_setup()

```
Led* led_setup (
    int number )
```

Setups a new instance of a [Led](#).

##### Parameters

<i>number</i>	Digital pin number associated to the <a href="#">Led</a> .
---------------	--

##### Returns

A pointer to [Led](#) structure.

#### 5.14.1.6 led\_switch\_light()

```
void led_switch_light (
    Led * led )
```

Toggles the instance [Led](#)'s light.

##### Parameters

<i>led</i>	<a href="#">Led</a> instance.
------------	-------------------------------

## 5.15 src/pins.c File Reference

```
#include "pins.h"
```

### Functions

- [uint8\\_t set\\_output](#) (uint8\_t number)  
*Sets a pin as a output, using its DDR.*
- [uint8\\_t set\\_pullup](#) (uint8\_t number)  
*Sets a pin as active, using its PORT.*
- [uint8\\_t toggle\\_pullup](#) (uint8\_t number)  
*Toggles a pin activity, using its PORT.*
- [uint8\\_t test\\_input](#) (uint8\_t number)  
*Tests if there is an input signal on a pin, using its PIN.*
- [uint8\\_t set\\_input](#) (uint8\_t number)  
*Sets a pin as an input, using its DDR.*
- [uint8\\_t clr\\_pullup](#) (uint8\_t number)  
*Sets a pin as inactive, using its PORT.*
- [uint8\\_t digital\\_write](#) (uint8\_t number, int value)  
*Writes an integer value to a pin, using its PORT.*

## Variables

- [Pin Pins \[\]](#)  
*Digital pin instances, common to all files.*
- [bytes Bytes \[\]](#)  
*Bytes instances.*

## 5.15.1 Function Documentation

### 5.15.1.1 clr\_pullup()

```
uint8_t clr_pullup (  
    uint8_t number )
```

Sets a pin as inactive, using its PORT.

#### Parameters

<i>number</i>	Digital pin number.
---------------	---------------------

### 5.15.1.2 digital\_write()

```
uint8_t digital_write (  
    uint8_t number,  
    int value )
```

Writes an integer value to a pin, using its PORT.

#### Parameters

<i>number</i>	Digital pin number.
---------------	---------------------

### 5.15.1.3 set\_input()

```
uint8_t set_input (  
    uint8_t number )
```

Sets a pin as an input, using its DDR.

**Parameters**

<i>number</i>	Digital pin number.
---------------	---------------------

**5.15.1.4 set\_output()**

```
uint8_t set_output (
    uint8_t number )
```

Sets a pin as a output, using its DDR.

**Parameters**

<i>number</i>	Digital pin number.
---------------	---------------------

**5.15.1.5 set\_pullup()**

```
uint8_t set_pullup (
    uint8_t number )
```

Sets a pin as active, using its PORT.

**Parameters**

<i>number</i>	Digital pin number.
---------------	---------------------

**5.15.1.6 test\_input()**

```
uint8_t test_input (
    uint8_t number )
```

Tests if there is an input signal on a pin, using its PIN.

**Parameters**

<i>number</i>	Digital pin number.
---------------	---------------------



### 5.15.1.7 toggle\_pullup()

```
uint8_t toggle_pullup (
    uint8_t number )
```

Toggles a pin activity, using its PORT.

#### Parameters

<i>number</i>	Digital pin number.
---------------	---------------------

## 5.15.2 Variable Documentation

### 5.15.2.1 Bytes

```
bytes Bytes[ ]
```

#### Initial value:

```
= {
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {WGM20, WGM21, WGM22, COM2B0, COM2B1, CS20, CS21, CS22, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {WGM00, WGM01, WGM02, COM0B0, COM0B1, CS00, CS01, CS02, 0},
    {WGM00, WGM01, WGM02, COM0A0, COM0A1, CS00, CS01, CS02, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {WGM10, WGM11, WGM12, COM1A0, COM1A1, CS10, CS11, CS12, WGM13},
    {WGM10, WGM11, WGM12, COM1B0, COM1B1, CS10, CS11, CS12, WGM13},
    {WGM20, WGM21, WGM22, COM2A0, COM2A1, CS20, CS21, CS22, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0}
}
```

Bytes instances.

### 5.15.2.2 Pins

```
Pin Pins[ ]
```

#### Initial value:

```
= {
    {PD0, &PORTD, &DDRD, &PIND, NULL, NULL, NULL, NULL},
    {PD1, &PORTD, &DDRD, &PIND, NULL, NULL, NULL, NULL},
    {PD2, &PORTD, &DDRD, &PIND, NULL, NULL, NULL, NULL},
    {PD3, &PORTD, &DDRD, &PIND, &TCCR2A, &TCCR2B, &OCR2A, &OCR2B},
    {PD4, &PORTD, &DDRD, &PIND, NULL, NULL, NULL, NULL},
    {PD5, &PORTD, &DDRD, &PIND, &TCCR0A, &TCCR0B, &OCR0A, &OCR0B},
    {PD6, &PORTD, &DDRD, &PIND, &TCCR0A, &TCCR0B, &OCR0A, NULL},
    {PD7, &PORTD, &DDRD, &PIND, NULL, NULL, NULL, NULL},
    {PB0, &PORTB, &DDRB, &PINB, NULL, NULL, NULL, NULL},
    {PB1, &PORTB, &DDRB, &PINB, &TCCR1A, &TCCR1B, (uint8_t*)&OCR1A, NULL},
    {PB2, &PORTB, &DDRB, &PINB, &TCCR1A, &TCCR1B, (uint8_t*)&OCR1A, (uint8_t*)&OCR1B},
    {PB3, &PORTB, &DDRB, &PINB, &TCCR2A, &TCCR2B, &OCR2A, NULL},
    {PB4, &PORTB, &DDRB, &PINB, NULL, NULL, NULL, NULL},
    {PB5, &PORTB, &DDRB, &PINB, NULL, NULL, NULL, NULL}
}
```

Digital pin instances, common to all files.

## 5.16 src/pwm.c File Reference

```
#include "pwm.h"
#include "pins.h"
```

### Functions

- float [lookup\\_prescalar](#) (float freq)  
*Specifies the prescalar based on frequency.*
- uint8\_t [lookup\\_cs](#) (float freq)  
*Specifies the CS control bits value based on frequency.*
- uint8\_t [lookup\\_cs2](#) (float freq)
- int [lookup\\_ocr](#) (float freq)  
*Specifies OCR value based on frequency.*
- void [sqrwave](#) (uint8\_t pin, float freq)  
*Generates square waves pulse in a pin based on frequency.*
- void [pwmwave](#) (uint8\_t pin, uint8\_t dutycycle)  
*Generates PWM waves in a pin based on a duty cycle.*

### 5.16.1 Function Documentation

#### 5.16.1.1 [lookup\\_cs\(\)](#)

```
uint8_t lookup_cs (
    float )
```

Specifies the CS control bits value based on frequency.

#### Parameters

<i>freq</i>	Frequency value (Hz)
-------------	----------------------

#### Returns

CS control bits value

#### 5.16.1.2 [lookup\\_cs2\(\)](#)

```
uint8_t lookup_cs2 (
    float freq )
```

#### 5.16.1.3 lookup\_ocr()

```
int lookup_ocr (
    float )
```

Specifies OCR value based on frequency.

##### Parameters

<i>freq</i>	Frequency value (Hz)
-------------	----------------------

##### Returns

OCR value

#### 5.16.1.4 lookup\_prescalar()

```
float lookup_prescalar (
    float )
```

Specifies the prescalar based on frequency.

##### Parameters

<i>freq</i>	Frequency value (Hz)
-------------	----------------------

##### Returns

Prescalar value

#### 5.16.1.5 pwmwave()

```
void pwmwave (
    uint8_t ,
    uint8_t )
```

Generates PWM waves in a pin based on a duty cycle.

##### Parameters

<i>pin</i>	Pin number
<i>dutycycle</i>	Duty cycle value (0 ~ 255)

#### 5.16.1.6 sqwave()

```
void sqwave (
    uint8_t ,
    float )
```

Generates square waves pulse in a pin based on frequency.

##### Parameters

<i>pin</i>	Pin number
<i>freq</i>	Frequency value (Hz)

## 5.17 src/README.md File Reference

## 5.18 src/supersonic.c File Reference

```
#include "supersonic.h"
#include "macros.h"
#include "pins.h"
#include <util/delay.h>
```

### Functions

- int [supersonic](#) (uint8\_t trig, uint8\_t echo)  
*Measures a distance based on a sonar.*

#### 5.18.1 Function Documentation

##### 5.18.1.1 supersonic()

```
int supersonic (
    uint8_t ,
    uint8_t )
```

Measures a distance based on a sonar.

##### Parameters

<i>trig</i>	Digital pin number associated to the Trigger pin.
<i>echo</i>	Digital pin number associated to the Echo pin.

### Returns

Measured distance (cm) between obstacle.

## 5.19 src/uart.c File Reference

```
#include "uart.h"
#include <avr/io.h>
#include <avr/sfr_defs.h>
#include <util/setbaud.h>
#include <stdio.h>
```

### Functions

- void [uart\\_init](#) (void)
- int [uart\\_putchar](#) (char c, FILE \*stream)
- int [uart\\_getchar](#) (FILE \*stream)

### Variables

- static FILE [uart\\_io](#) = FDEV\_SETUP\_STREAM([uart\\_putchar](#), [uart\\_getchar](#), \_FDEV\_SETUP\_RW)

## 5.19.1 Function Documentation

### 5.19.1.1 [uart\\_getchar\(\)](#)

```
int uart_getchar (
    FILE * stream )
```

### 5.19.1.2 [uart\\_init\(\)](#)

```
void uart_init (
    void )
```

### 5.19.1.3 [uart\\_putchar\(\)](#)

```
int uart_putchar (
    char c,
    FILE * stream )
```

## 5.19.2 Variable Documentation

### 5.19.2.1 [uart\\_io](#)

```
FILE uart_io = FDEV_SETUP_STREAM(uart\_putchar, uart\_getchar, _FDEV_SETUP_RW) [static]
```



# Index

- `_DDR`
    - Pin, [14](#)
  - `_OCRA`
    - Pin, [14](#)
  - `_OCRB`
    - Pin, [14](#)
  - `_PIN`
    - Pin, [14](#)
  - `_PORT`
    - Pin, [14](#)
  - `_TCCA`
    - Pin, [14](#)
  - `_TCCB`
    - Pin, [15](#)
- a
  - BCD, [8](#)
- b
  - BCD, [8](#)
- BAUD
  - uart.h, [57](#)
- BCD, [7](#)
  - a, [8](#)
  - b, [8](#)
  - c, [8](#)
  - d, [8](#)
  - delay, [8](#)
  - dp, [8](#)
  - e, [9](#)
  - f, [9](#)
  - g, [9](#)
  - loop, [9](#)
- bcd.c
  - bcd\_clear, [17](#)
  - bcd\_delay, [18](#)
  - bcd\_free, [18](#)
  - bcd\_set\_loop, [18](#)
  - bcd\_setup, [18](#)
  - bcd\_show\_char, [19](#)
  - bcd\_show\_string, [19](#)
- bcd.h
  - bcd\_clear, [24](#)
  - bcd\_delay, [24](#)
  - bcd\_free, [24](#)
  - bcd\_set\_loop, [25](#)
  - bcd\_setup, [25](#)
  - bcd\_show\_char, [26](#)
  - bcd\_show\_string, [26](#)
- bcd\_clear
  - bcd.c, [17](#)
  - bcd.h, [24](#)
- bcd\_delay
  - bcd.c, [18](#)
  - bcd.h, [24](#)
- bcd\_free
  - bcd.c, [18](#)
  - bcd.h, [24](#)
- bcd\_set\_loop
  - bcd.c, [18](#)
  - bcd.h, [25](#)
- bcd\_setup
  - bcd.c, [18](#)
  - bcd.h, [25](#)
- bcd\_seven.h
  - rep\_a, [29](#)
  - rep\_b, [30](#)
  - rep\_c, [30](#)
  - rep\_d, [30](#)
  - rep\_dp, [30](#)
  - rep\_e, [30](#)
  - rep\_f, [30](#)
  - rep\_g, [31](#)
  - SevenSegmentASCII, [31](#)
  - sseg\_0, [31](#)
  - sseg\_1, [31](#)
  - sseg\_2, [31](#)
  - sseg\_3, [32](#)
  - sseg\_4, [32](#)
  - sseg\_5, [32](#)
  - sseg\_6, [32](#)
  - sseg\_7, [32](#)
  - sseg\_8, [32](#)
  - sseg\_9, [33](#)
  - sseg\_A, [33](#)
  - sseg\_a, [33](#)
  - sseg\_B, [33](#)
  - sseg\_b, [33](#)
  - sseg\_C, [33](#)
  - sseg\_c, [34](#)
  - sseg\_D, [34](#)
  - sseg\_d, [34](#)
  - sseg\_E, [34](#)
  - sseg\_e, [34](#)
  - sseg\_F, [34](#)
  - sseg\_f, [35](#)
  - sseg\_G, [35](#)
  - sseg\_g, [35](#)
  - sseg\_H, [35](#)

- sseg\_h, 35
- sseg\_l, 35
- sseg\_i, 36
- sseg\_J, 36
- sseg\_j, 36
- sseg\_K, 36
- sseg\_k, 36
- sseg\_L, 36
- sseg\_l, 37
- sseg\_M, 37
- sseg\_m, 37
- sseg\_N, 37
- sseg\_n, 37
- sseg\_O, 37
- sseg\_o, 38
- sseg\_P, 38
- sseg\_p, 38
- sseg\_Q, 38
- sseg\_q, 38
- sseg\_R, 38
- sseg\_r, 39
- sseg\_S, 39
- sseg\_s, 39
- sseg\_T, 39
- sseg\_t, 39
- sseg\_U, 39
- sseg\_u, 40
- sseg\_V, 40
- sseg\_v, 40
- sseg\_W, 40
- sseg\_w, 40
- sseg\_X, 40
- sseg\_x, 41
- sseg\_Y, 41
- sseg\_y, 41
- sseg\_Z, 41
- sseg\_z, 41
- bcd\_show\_char
  - bcd.c, 19
  - bcd.h, 26
- bcd\_show\_string
  - bcd.c, 19
  - bcd.h, 26
- btn\_fired
  - button.c, 20
  - button.h, 43
- btn\_pressed
  - button.c, 21
  - button.h, 43
- btn\_register\_fn
  - button.c, 21
  - button.h, 43
- btn\_set\_mode
  - button.c, 21
  - button.h, 44
- btn\_setup
  - button.c, 21
  - button.h, 44
- Button, 9
  - fn, 10
  - mode, 10
  - pin\_number, 10
- button.c
  - btn\_fired, 20
  - btn\_pressed, 21
  - btn\_register\_fn, 21
  - btn\_set\_mode, 21
  - btn\_setup, 21
- button.h
  - btn\_fired, 43
  - btn\_pressed, 43
  - btn\_register\_fn, 43
  - btn\_set\_mode, 44
  - btn\_setup, 44
  - Button\_Mode, 42
  - button\_mode, 42
- Button\_Mode
  - button.h, 42
- button\_mode
  - button.h, 42
- Bytes
  - pins.c, 63
  - pins.h, 53
- bytes, 11
  - COM0, 11
  - COM1, 11
  - CS0, 11
  - CS1, 11
  - CS2, 11
  - WGM0, 12
  - WGM1, 12
  - WGM2, 12
  - WGM3, 12
- c
  - BCD, 8
- COM0
  - bytes, 11
- COM1
  - bytes, 11
- CS0
  - bytes, 11
- CS1
  - bytes, 11
- CS2
  - bytes, 11
- clr\_bit
  - macros.h, 49
- clr\_pullup
  - pins.c, 61
  - pins.h, 51
- comp\_masks
  - macros.h, 49
- d
  - BCD, 8
- delay



- BCD, [8](#)
- delay.c
  - delay\_1ms, [22](#)
  - delay\_1us, [22](#)
  - delay\_ms, [23](#)
  - delay\_us, [23](#)
- delay.h
  - delay\_1ms, [45](#)
  - delay\_1us, [45](#)
  - delay\_ms, [45](#)
  - delay\_us, [45](#)
- delay\_1ms
  - delay.c, [22](#)
  - delay.h, [45](#)
- delay\_1us
  - delay.c, [22](#)
  - delay.h, [45](#)
- delay\_ms
  - delay.c, [23](#)
  - delay.h, [45](#)
- delay\_us
  - delay.c, [23](#)
  - delay.h, [45](#)
- digital\_write
  - pins.c, [61](#)
  - pins.h, [52](#)
- dp
  - BCD, [8](#)
- e
  - BCD, [9](#)
- f
  - BCD, [9](#)
- F\_CPU
  - pins.h, [51](#)
  - uart.h, [57](#)
- fn
  - Button, [10](#)
- g
  - BCD, [9](#)
- HIGH
  - pins.h, [51](#)
- LOW
  - pins.h, [51](#)
- Led, [12](#)
  - pin\_number, [13](#)
  - state, [13](#)
- led.c
  - led\_blink, [58](#)
  - led\_free, [59](#)
  - led\_kill\_light, [59](#)
  - led\_make\_light, [59](#)
  - led\_setup, [59](#)
  - led\_switch\_light, [60](#)
- led.h
  - Led\_State, [46](#)
  - led\_blink, [47](#)
  - led\_free, [47](#)
  - led\_kill\_light, [48](#)
  - led\_make\_light, [48](#)
  - led\_setup, [48](#)
  - led\_state, [47](#)
  - led\_switch\_light, [48](#)
- Led\_State
  - led.h, [46](#)
- led\_blink
  - led.c, [58](#)
  - led.h, [47](#)
- led\_free
  - led.c, [59](#)
  - led.h, [47](#)
- led\_kill\_light
  - led.c, [59](#)
  - led.h, [48](#)
- led\_make\_light
  - led.c, [59](#)
  - led.h, [48](#)
- led\_setup
  - led.c, [59](#)
  - led.h, [48](#)
- led\_state
  - led.h, [47](#)
- led\_switch\_light
  - led.c, [60](#)
  - led.h, [48](#)
- lookup\_cs
  - pwm.c, [64](#)
  - pwm.h, [54](#)
- lookup\_cs2
  - pwm.c, [64](#)
  - pwm.h, [54](#)
- lookup\_ocr
  - pwm.c, [64](#)
  - pwm.h, [55](#)
- lookup\_prescalar
  - pwm.c, [65](#)
  - pwm.h, [55](#)
- loop
  - BCD, [9](#)
- MACROS
  - macros.h, [49](#)
- macros.h
  - clr\_bit, [49](#)
  - comp\_masks, [49](#)
  - MACROS, [49](#)
  - set\_bit, [49](#)
  - swt\_bit, [50](#)
  - tst\_bit, [50](#)
- mode
  - Button, [10](#)
- P
  - Pin, [15](#)

- Pin, [13](#)
  - [\\_DDR, 14](#)
  - [\\_OCRA, 14](#)
  - [\\_OCRB, 14](#)
  - [\\_PIN, 14](#)
  - [\\_PORT, 14](#)
  - [\\_TCCA, 14](#)
  - [\\_TCCB, 15](#)
  - [P, 15](#)
- pin\_number
  - [Button, 10](#)
  - [Led, 13](#)
- Pins
  - [pins.c, 63](#)
  - [pins.h, 53](#)
- pins.c
  - [Bytes, 63](#)
  - [clr\\_pullup, 61](#)
  - [digital\\_write, 61](#)
  - [Pins, 63](#)
  - [set\\_input, 61](#)
  - [set\\_output, 62](#)
  - [set\\_pullup, 62](#)
  - [test\\_input, 62](#)
  - [toggle\\_pullup, 62](#)
- pins.h
  - [Bytes, 53](#)
  - [clr\\_pullup, 51](#)
  - [digital\\_write, 52](#)
  - [F\\_CPU, 51](#)
  - [HIGH, 51](#)
  - [LOW, 51](#)
  - [Pins, 53](#)
  - [set\\_input, 52](#)
  - [set\\_output, 52](#)
  - [set\\_pullup, 52](#)
  - [test\\_input, 53](#)
  - [toggle\\_pullup, 53](#)
- pwm.c
  - [lookup\\_cs, 64](#)
  - [lookup\\_cs2, 64](#)
  - [lookup\\_ocr, 64](#)
  - [lookup\\_prescaler, 65](#)
  - [pwmwave, 65](#)
  - [sqrwave, 65](#)
- pwm.h
  - [lookup\\_cs, 54](#)
  - [lookup\\_cs2, 54](#)
  - [lookup\\_ocr, 55](#)
  - [lookup\\_prescaler, 55](#)
  - [pwmwave, 55](#)
  - [sqrwave, 56](#)
- pwmwave
  - [pwm.c, 65](#)
  - [pwm.h, 55](#)
- rep\_a
  - [bcd\\_seven.h, 29](#)
- rep\_b
  - [bcd\\_seven.h, 30](#)
- rep\_c
  - [bcd\\_seven.h, 30](#)
- rep\_d
  - [bcd\\_seven.h, 30](#)
- rep\_dp
  - [bcd\\_seven.h, 30](#)
- rep\_e
  - [bcd\\_seven.h, 30](#)
- rep\_f
  - [bcd\\_seven.h, 30](#)
- rep\_g
  - [bcd\\_seven.h, 31](#)
- set\_bit
  - [macros.h, 49](#)
- set\_input
  - [pins.c, 61](#)
  - [pins.h, 52](#)
- set\_output
  - [pins.c, 62](#)
  - [pins.h, 52](#)
- set\_pullup
  - [pins.c, 62](#)
  - [pins.h, 52](#)
- SevenSegmentASCII
  - [bcd\\_seven.h, 31](#)
- sqrwave
  - [pwm.c, 65](#)
  - [pwm.h, 56](#)
- src/README.md, [66](#)
- src/bcd.c, [17](#)
- src/button.c, [20](#)
- src/delay.c, [22](#)
- src/include/bcd.h, [23](#)
- src/include/bcd\_seven.h, [26](#)
- src/include/button.h, [42](#)
- src/include/delay.h, [44](#)
- src/include/led.h, [46](#)
- src/include/macros.h, [49](#)
- src/include/pins.h, [50](#)
- src/include/pwm.h, [54](#)
- src/include/supersonic.h, [56](#)
- src/include/uart.h, [57](#)
- src/led.c, [58](#)
- src/pins.c, [60](#)
- src/pwm.c, [64](#)
- src/supersonic.c, [66](#)
- src/uart.c, [67](#)
- sseg\_0
  - [bcd\\_seven.h, 31](#)
- sseg\_1
  - [bcd\\_seven.h, 31](#)
- sseg\_2
  - [bcd\\_seven.h, 31](#)
- sseg\_3
  - [bcd\\_seven.h, 32](#)
- sseg\_4
  - [bcd\\_seven.h, 32](#)

sseg\_5  
    bcd\_seven.h, [32](#)  
sseg\_6  
    bcd\_seven.h, [32](#)  
sseg\_7  
    bcd\_seven.h, [32](#)  
sseg\_8  
    bcd\_seven.h, [32](#)  
sseg\_9  
    bcd\_seven.h, [33](#)  
sseg\_A  
    bcd\_seven.h, [33](#)  
sseg\_a  
    bcd\_seven.h, [33](#)  
sseg\_B  
    bcd\_seven.h, [33](#)  
sseg\_b  
    bcd\_seven.h, [33](#)  
sseg\_C  
    bcd\_seven.h, [33](#)  
sseg\_c  
    bcd\_seven.h, [34](#)  
sseg\_D  
    bcd\_seven.h, [34](#)  
sseg\_d  
    bcd\_seven.h, [34](#)  
sseg\_E  
    bcd\_seven.h, [34](#)  
sseg\_e  
    bcd\_seven.h, [34](#)  
sseg\_F  
    bcd\_seven.h, [34](#)  
sseg\_f  
    bcd\_seven.h, [35](#)  
sseg\_G  
    bcd\_seven.h, [35](#)  
sseg\_g  
    bcd\_seven.h, [35](#)  
sseg\_H  
    bcd\_seven.h, [35](#)  
sseg\_h  
    bcd\_seven.h, [35](#)  
sseg\_I  
    bcd\_seven.h, [35](#)  
sseg\_i  
    bcd\_seven.h, [36](#)  
sseg\_J  
    bcd\_seven.h, [36](#)  
sseg\_j  
    bcd\_seven.h, [36](#)  
sseg\_K  
    bcd\_seven.h, [36](#)  
sseg\_k  
    bcd\_seven.h, [36](#)  
sseg\_L  
    bcd\_seven.h, [36](#)  
sseg\_l  
    bcd\_seven.h, [37](#)  
sseg\_M  
    bcd\_seven.h, [37](#)  
sseg\_m  
    bcd\_seven.h, [37](#)  
sseg\_N  
    bcd\_seven.h, [37](#)  
sseg\_n  
    bcd\_seven.h, [37](#)  
sseg\_O  
    bcd\_seven.h, [37](#)  
sseg\_o  
    bcd\_seven.h, [38](#)  
sseg\_P  
    bcd\_seven.h, [38](#)  
sseg\_p  
    bcd\_seven.h, [38](#)  
sseg\_Q  
    bcd\_seven.h, [38](#)  
sseg\_q  
    bcd\_seven.h, [38](#)  
sseg\_R  
    bcd\_seven.h, [38](#)  
sseg\_r  
    bcd\_seven.h, [39](#)  
sseg\_S  
    bcd\_seven.h, [39](#)  
sseg\_s  
    bcd\_seven.h, [39](#)  
sseg\_T  
    bcd\_seven.h, [39](#)  
sseg\_t  
    bcd\_seven.h, [39](#)  
sseg\_U  
    bcd\_seven.h, [39](#)  
sseg\_u  
    bcd\_seven.h, [40](#)  
sseg\_V  
    bcd\_seven.h, [40](#)  
sseg\_v  
    bcd\_seven.h, [40](#)  
sseg\_W  
    bcd\_seven.h, [40](#)  
sseg\_w  
    bcd\_seven.h, [40](#)  
sseg\_X  
    bcd\_seven.h, [40](#)  
sseg\_x  
    bcd\_seven.h, [41](#)  
sseg\_Y  
    bcd\_seven.h, [41](#)  
sseg\_y  
    bcd\_seven.h, [41](#)  
sseg\_Z  
    bcd\_seven.h, [41](#)  
sseg\_z  
    bcd\_seven.h, [41](#)  
state  
    Led, [13](#)

- supersonic
  - supersonic.c, [66](#)
  - supersonic.h, [56](#)
- supersonic.c
  - supersonic, [66](#)
- supersonic.h
  - supersonic, [56](#)
- swt\_bit
  - macros.h, [50](#)
- test\_input
  - pins.c, [62](#)
  - pins.h, [53](#)
- toggle\_pullup
  - pins.c, [62](#)
  - pins.h, [53](#)
- tst\_bit
  - macros.h, [50](#)
- uart.c
  - uart\_getchar, [67](#)
  - uart\_init, [67](#)
  - uart\_io, [67](#)
  - uart\_putchar, [67](#)
- uart.h
  - BAUD, [57](#)
  - F\_CPU, [57](#)
  - uart\_getchar, [57](#)
  - uart\_init, [58](#)
  - uart\_putchar, [58](#)
- uart\_getchar
  - uart.c, [67](#)
  - uart.h, [57](#)
- uart\_init
  - uart.c, [67](#)
  - uart.h, [58](#)
- uart\_io
  - uart.c, [67](#)
- uart\_putchar
  - uart.c, [67](#)
  - uart.h, [58](#)
- WGM0
  - bytes, [12](#)
- WGM1
  - bytes, [12](#)
- WGM2
  - bytes, [12](#)
- WGM3
  - bytes, [12](#)