lib-arduino

Generated by Doxygen 1.8.15

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 bilblioteca 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:

```
avrdude
gcc-avr
```

Utilização

Crie um arquivo fonte *main.c* no diretorio *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:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 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

display	Display instance.

5.1.1.2 bcd_delay()

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

Freezes a display for a presetted delay.

Parameters

display	Display instance.

5.1.1.3 bcd_free()

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

Destroys a display instance.

Parameters

display	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

display	Display instance.
loop	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

a	Digital pin where the display pin 'a' is plugged.
b	Digital pin where the display pin 'b' is plugged.
c	Digital pin where the display pin 'c' is plugged.
d	Digital pin where the display pin 'd' is plugged.
e	Digital pin where the display pin 'e' is plugged.
f	Digital pin where the display pin 'f' is plugged.
g	Digital pin where the display pin 'g' is plugged.
dp	Digital pin where the dot pin of display is plugged.
delay	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

display	Display instance.
character	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

display	Display instance.
str	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

b	Button instance.

5.2.1.2 btn_pressed()

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

Verifies if a button is pressed at moment.

Parameters

b	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

b	Button instance.
function	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

b	Button instance.
mode	New button mode.

5.2.1.5 btn_setup()

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

Setups a new instance of a Button.

Parameters

number	Digital pin number associated to the Button.

Returns

A pointer to Button structure.

5.3 src/delay.c File Reference

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

Functions

- void delay_1us ()

  Delays for, approximately, 1 microssecond.
- void delay_1ms ()

  Delays for, approximately, 1 millissecond.
- void delay_us (uint32_t time)

  Delays for, approximately, a specified time, in microsseconds.
- 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 millissecond.

5.3.1.2 delay_1us()

```
void delay_1us (
             void  )
```

Delays for, approximately, 1 microssecond.

5.3.1.3 delay_ms()

```
void delay_ms (
             uint32_t  )
```

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

Parameters

time	Time for delay (in milliseconds)

5.3.1.4 delay_us()

```
void delay_us (
             uint32_t  )
```

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

Parameters

time	Time for delay (in microsseconds)

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

display	Display instance.

5.4.1.2 bcd_delay()

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

Freezes a display for a presetted delay.

Parameters

display	Display instance.

5.4.1.3 bcd_free()

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

Destroys a display instance.

Parameters

display	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

display	Display instance.
loop	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

a	Digital pin where the display pin 'a' is plugged.
b	Digital pin where the display pin 'b' is plugged.
c	Digital pin where the display pin 'c' is plugged.
d	Digital pin where the display pin 'd' is plugged.
e	Digital pin where the display pin 'e' is plugged.

Parameters

f	Digital pin where the display pin 'f' is plugged.
g	Digital pin where the display pin 'g' is plugged.
dp	Digital pin where the dot pin of display is plugged.
delay	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

display	Display instance.
character	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

display	Display instance.
str	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_l

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

Specifies pins to write 'l'.

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

b	Button instance.

5.6.3.2 btn_pressed()

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

Verifies if a button is pressed at moment.

Parameters

b	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

b	Button instance.
function	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

b	Button instance.
mode	New button mode.

5.6.3.5 btn_setup()

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

Setups a new instance of a Button.

Parameters

number	Digital pin number associated to the Button.

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 microssecond.

- void delay_1ms (void)

  Delays for, approximately, 1 millissecond.

- void delay_us (uint32_t)

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

- 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 millissecond.

5.7.1.2 delay_1us()

```
void delay_1us (
             void   )
```

Delays for, approximately, 1 microssecond.

5.7.1.3 delay_ms()

```
void delay_ms (
             uint32_t   )
```

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

Parameters

time	Time for delay (in milliseconds)

5.7.1.4 delay_us()

```
void delay_us (
             uint32_t   )
```

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

Parameters

time	Time for delay (in microsseconds)

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	Led's light is on.
Off	Led'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

led	Led instance.
time_ms	Time interval to keep Led's light on and off.

5.8.3.2 led_free()

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

Destroys a Led instance.

Parameters

led	Led instance.

5.8.3.3 led_kill_light()

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

Turns off the instance Led's light.

Parameters

led	Led instance.

5.8.3.4 led_make_light()

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

Turns on the instance Led's light.

Parameters

led	Led instance.

5.8.3.5 led_setup()

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

Setups a new instance of a Led.

Parameters

number	Digital pin number associated to the Led.

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

led	Led 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

number	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

number	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

number	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

number	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

number	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

number	Digital pin number.

5.10.2.7 toggle_pullup()

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

Toggles a pin activity, using its PORT.

Parameters

number	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

freq	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

freq	Frequency value (Hz)

Returns

OCR value

5.11.1.4 lookup_prescalar()

```
float lookup_prescalar (
            float  )
```

Specifies the prescalar based on frequency.

Parameters

freq	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

pin	Pin number
dutycycle	Duty cycle value (0 ∼ 255)

5.11.1.6 sqrwave()

```
void sqrwave (
            uint8_t ,
            float  )
```

Generates square waves pulse in a pin based on frequency.

Parameters

pin	Pin number
freq	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

trig	Digital pin number associated to the Trigger pin.
echo	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

led	Led instance.
time_ms	Time interval to keep Led's light on and off.

5.14.1.2 led_free()

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

Destroys a Led instance.

Parameters

led	Led instance.

5.14.1.3 led_kill_light()

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

Turns off the instance Led's light.

Parameters

led	Led instance.

5.14.1.4 led_make_light()

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

Turns on the instance Led's light.

Parameters

led	Led instance.

5.14.1.5 led_setup()

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

Setups a new instance of a Led.

Parameters

number	Digital pin number associated to the Led.

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

led	Led 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

number	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

number	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

number	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

number	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

number	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

number	Digital pin number.

5.15.1.7 toggle_pullup()

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

Toggles a pin activity, using its PORT.

Parameters

number	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},
    {WGM20, WGM21, WGM22, COM2B0, COM2B1, CS20, CS21, CS22, 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

freq	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

freq	Frequency value (Hz)

Returns

OCR value

5.16.1.4 lookup_prescalar()

```
float lookup_prescalar (
            float  )
```

Specifies the prescalar based on frequency.

Parameters

freq	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

pin	Pin number
dutycycle	Duty cycle value (0 $\sim$ 255)

5.16.1.6 sqrwave()

```
void sqrwave (
            uint8_t ,
            float  )
```

Generates square waves pulse in a pin based on frequency.

Parameters

pin	Pin number
freq	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

trig	Digital pin number associated to the Trigger pin.
echo	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