SM_project

Generated by Doxygen 1.9.3

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 HD44780 Struct Reference

`#include <HD44780.h>`

Data Fields

- GPIO_TypeDef * rs_gpio
- GPIO_TypeDef * rw_gpio
- GPIO_TypeDef * en_gpio
- GPIO_TypeDef * d0_gpio
- GPIO_TypeDef * d1_gpio
- GPIO_TypeDef * d2_gpio
- GPIO_TypeDef * d3_gpio
- GPIO_TypeDef * d4_gpio
- GPIO_TypeDef * d5_gpio
- GPIO_TypeDef * d6_gpio
- GPIO_TypeDef * d7_gpio
- uint16_t rs_pin
- uint16_t rw_pin
- uint16_t en_pin
- uint16_t d0_pin
- uint16_t d1_pin
- uint16_t d2_pin
- uint16_t d3_pin
- uint16_t d4_pin
- uint16_t d5_pin
- uint16_t d6_pin
- uint16_t d7_pin
- bool interface_8_bit
- bool single_line
- bool font_5x10

3.1.1 Detailed Description

HD44780 controller instance. Contains all the information on the hardware configuration of the controller, and some required initialization settings.

3.1.2 Field Documentation

3.1.2.1 d0_gpio

```
GPIO_TypeDef* d0_gpio
```

GPIO port of the mcu pin connected to the controller's D0 line.

3.1.2.2 d0_pin

```
uint16_t d0_pin
```

Pin number of the mcu pin connected to the controller's D0 line.

3.1.2.3 d1_gpio

```
GPIO_TypeDef* d1_gpio
```

GPIO port of the mcu pin connected to the controller's D1 line.

3.1.2.4 d1_pin

```
uint16_t d1_pin
```

Pin number of the mcu pin connected to the controller's D1 line.

3.1.2.5 d2_gpio

```
GPIO_TypeDef* d2_gpio
```

GPIO port of the mcu pin connected to the controller's D2 line.

3.1.2.6 d2_pin

```
uint16_t d2_pin
```

Pin number of the mcu pin connected to the controller's D2 line.

3.1.2.7 d3_gpio

```
GPIO_TypeDef* d3_gpio
```

GPIO port of the mcu pin connected to the controller's D3 line.

3.1.2.8 d3_pin

```
uint16_t d3_pin
```

Pin number of the mcu pin connected to the controller's D3 line.

3.1.2.9 d4_gpio

```
GPIO_TypeDef* d4_gpio
```

GPIO port of the mcu pin connected to the controller's D4 line.

3.1.2.10 d4_pin

```
uint16_t d4_pin
```

Pin number of the mcu pin connected to the controller's D4 line.

3.1.2.11 d5_gpio

```
GPIO_TypeDef* d5_gpio
```

GPIO port of the mcu pin connected to the controller's D5 line.

3.1.2.12 d5_pin

```
uint16_t d5_pin
```

Pin number of the mcu pin connected to the controller's D5 line.

3.1.2.13 d6_gpio

```
GPIO_TypeDef* d6_gpio
```

GPIO port of the mcu pin connected to the controller's D6 line.

3.1.2.14 d6_pin

```
uint16_t d6_pin
```

Pin number of the mcu pin connected to the controller's D6 line.

3.1.2.15 d7_gpio

```
GPIO_TypeDef* d7_gpio
```

GPIO port of the mcu pin connected to the controller's D7 line.

3.1.2.16 d7_pin

```
uint16_t d7_pin
```

Pin number of the mcu pin connected to the controller's D7 line.

3.1.2.17 en_gpio

```
GPIO_TypeDef* en_gpio
```

GPIO port of the mcu pin connected to the controller's EN line.

3.1.2.18 en_pin

```
uint16_t en_pin
```

Pin number of the mcu pin connected to the controller's EN line.

3.1.2.19 font_5x10

```
bool font_5x10
```

Use the 5x10 dots character font instead of the default 5x8 dots font.

Warning

The 5x10 dots font only supports single line operation (single_line = true).

3.1.2.20 interface_8_bit

```
bool interface_8_bit
```

Use 8 physical data lines (DB7-DB0) for communication with the controller instead of the default 4 lines (DB7-DB4).

3.1.2.21 rs_gpio

```
GPIO_TypeDef* rs_gpio
```

GPIO port of the mcu pin connected to the controller's RS line.

3.1.2.22 rs_pin

```
uint16_t rs_pin
```

Pin number of the mcu pin connected to the controller's RS line.

3.1.2.23 rw_gpio

```
GPIO_TypeDef* rw_gpio
```

GPIO port of the mcu pin connected to the controller's RW line.

3.1.2.24 rw_pin

```
uint16_t rw_pin
```

Pin number of the mcu pin connected to the controller's RW line.

3.1.2.25 single_line

```
bool single_line
```

Display a single taller line on the display instead of the default 2 lines.

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

- HD44780.h

3.2 HD44780_Config Struct Reference

```
#include <HD44780.h>
```

Data Fields

- bool disable_display
- bool enable_cursor
- bool enable_blink
- bool shift_display
- bool shift_rtl

3.2.1 Detailed Description

HD44780 controller configuration. Use in conjunction with HD44780_configure() to enable/disable the different features of the lcd.

3.2.2 Field Documentation

3.2.2.1 disable_display

```
bool disable_display
```

Disable the display. Data can still be written with the lcd unpowered and displayed at a later moment.

3.2.2.2 enable_blink

```
bool enable_blink
```

Make the character indicated by the cursor blink. The blinking is displayed as switching between all blank dots and displayed character.

3.2.2.3 enable_cursor

```
bool enable_cursor
```

Make the cursor visible. The cursor is displayed using 5 dots in the 8th line for 5×8 dots font and in the 11th line for 5×10 dots font.

3.2.2.4 shift_display

```
bool shift_display
```

Shift the display when data is entered instead of shifting only the cursor.

3.2.2.5 shift_rtl

```
bool shift_rtl
```

Shift the display (depending on the shift_display parameter) and/or cursor in right to left direction instead of left to right when a character is entered.

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

- HD44780.h

3.3 two_position_t Struct Reference

```
#include <regulator.h>
```

Data Fields

- float32_t H
- float32_t u_min
- float32_t u_max
- float32_t u_value

3.3.1 Field Documentation

3.3.1.1 H

```
float32_t H
```

3.3.1.2 u_max

```
float32_t u_max
```

3.3.1.3 u_min

```
float32_t u_min
```

3.3.1.4 u_value

```
float32_t u_value
```

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

- regulator.h

Chapter 4

File Documentation

4.1 HD44780.c File Reference

```
#include "HD44780.h"
```

Macros

- #define delay_us(us) delay_ns(us ∗ 1000)
- #define delay_ms(ms) delay_ns(ms ∗ 1000000)

Functions

- void HD44780_init (const HD44780 ∗lcd)
- void HD44780_configure (const HD44780 ∗lcd, const HD44780_Config ∗config)
- void HD44780_clear (const HD44780 ∗lcd)
- void HD44780_return_home (const HD44780 ∗lcd)
- void HD44780_cursor_to (const HD44780 ∗lcd, uint8_t column, uint8_t row)
- void HD44780_shift_display (const HD44780 ∗lcd, int8_t n)
- void HD44780_create_symbol (const HD44780 ∗lcd, uint8_t address, bool font_5x10, const uint8_t symbol[ ])
- void HD44780_put_char (const HD44780 ∗lcd, uint8_t chr)
- void HD44780_put_str (const HD44780 ∗lcd, const char ∗str)

4.1.1 Macro Definition Documentation

4.1.1.1 delay_ms

```
#define delay_ms(
             ms ) delay_ns(ms * 1000000)
```

Halt the program execution for the desired number of milliseconds.

4.1.1.2 delay_us

```
#define delay_us(
              us ) delay_ns(us * 1000)
```

Halt the program execution for the desired number of microseconds.

4.1.2 Function Documentation

4.1.2.1 HD44780_clear()

```
void HD44780_clear (
             const HD44780 * lcd )
```

Clear the display and move the cursor to position 0 of the first line.

Parameters

lcd	Controller instance.

4.1.2.2 HD44780_configure()

```
void HD44780_configure (
             const HD44780 * lcd,
             const HD44780_Config * config )
```

Update the configuration of the controller.

Parameters

lcd	Controller instance.
config	New controller configuration.

4.1.2.3 HD44780_create_symbol()

```
void HD44780_create_symbol (
             const HD44780 * lcd,
             uint8_t address,
             bool font_5x10,
             const uint8_t symbol[] )
```

Create a user defined character to display in the LCD. The controller memory can store up to 8 5x8 symbols, and up to 4 5x10 symbols.

Warning

5x10 symbols need 2 CGRAM slots, so after defining a 5x10 symbol at address n, the next symbol should be defined at address n+2.

Parameters

lcd	Controller instance.
address	ASCII code of the new symbol, must be in the range from 0 to 7 inclusive.
font_5x10	Whetеher the new symbol will be a 5x10 character.
symbol	Array of 5 bit values where each bit will determine whether the corresponding pixel is lit up in its corresponding row.

4.1.2.4 HD44780_cursor_to()

```
void HD44780_cursor_to (
            const HD44780 * lcd,
            uint8_t column,
            uint8_t row )
```

Move the cursor to the desired position.

Parameters

lcd	Controller instance.
column	Index of the desired cursor position in the line. Must be less than 0x50 in single line mode, and less than 0x28 in two lines mode, or the cursor will wrap to the next line causing undefined behaviour.
row	Index of the desired row. Must be 0 if the controller is configured for single line mode, and 0 or 1 when the controller is in two lines mode.

4.1.2.5 HD44780_init()

```
void HD44780_init (
            const HD44780 * lcd )
```

Initialize the necessary hardware peripherals, then configure the controller itself. The initial configuration will be the same as calling HD44780_configure() with all the config flags set to false.

Parameters

lcd	Controller instance.

4.1.2.6 HD44780_put_char()

```
void HD44780_put_char (
            const HD44780 * lcd,
            uint8_t chr )
```

Write a single character to the lcd, then advance the cursor. When the character is '\n' the cursor will advance to the next line, wrapping around from last to first. When the character is '\t' 4 spaces will be written to the display.

Parameters

lcd	Controller instance.
chr	Character to be printed to the lcd.

4.1.2.7 HD44780_put_str()

```
void HD44780_put_str (
            const HD44780 * lcd,
            const char * str )
```

Write a string to the lcd, then advance the cursor. The same considerations for special characters from HD44780_put_char() apply to this function.

Warning

The string must be null terminated.

Parameters

lcd	Controller instance.
str	String to be printed to the lcd.

4.1.2.8 HD44780_return_home()

```
void HD44780_return_home (
            const HD44780 * lcd )
```

Reset display shift to the initial position and move the cursor to position 0 of the first line.

Parameters

lcd	Controller instance.

4.1.2.9 HD44780_shift_display()

```
void HD44780_shift_display (
            const HD44780 * lcd,
            int8_t n )
```

Shift the contents of the display right or left by n positions. The first and second line will shift at the same time.

Note

The execution time of this function will increase linearly with the numbers of positions shifted (∼37us / position shifted).

Parameters

lcd	Controller instance.
n	Number of positions to shift. When the value is positive the diplay will shift left to right, when negative the shift operation will advance right to left.

4.2 HD44780.h File Reference

```
#include <stdbool.h>
#include <stdlib.h>
#include "stm32f2xx_hal.h"
```

Data Structures

- struct HD44780
- struct HD44780_Config

Functions

- void HD44780_init (const HD44780 ∗lcd)
- void HD44780_configure (const HD44780 ∗lcd, const HD44780_Config ∗config)
- void HD44780_clear (const HD44780 ∗lcd)
- void HD44780_return_home (const HD44780 ∗lcd)
- void HD44780_cursor_to (const HD44780 ∗lcd, uint8_t column, uint8_t row)
- void HD44780_shift_display (const HD44780 ∗lcd, int8_t n)
- void HD44780_create_symbol (const HD44780 ∗lcd, uint8_t address, bool font_5x10, const uint8_t symbol[ ])
- void HD44780_put_char (const HD44780 ∗lcd, uint8_t chr)
- void HD44780_put_str (const HD44780 ∗lcd, const char ∗str)

4.2.1 Function Documentation

4.2.1.1 HD44780_clear()

```
void HD44780_clear (
            const HD44780 * lcd )
```

Clear the display and move the cursor to position 0 of the first line.

Parameters

lcd	Controller instance.

4.2.1.2 HD44780_configure()

```
void HD44780_configure (
            const HD44780 * lcd,
            const HD44780_Config * config )
```

Update the configuration of the controller.

Parameters

lcd	Controller instance.
config	New controller configuration.

4.2.1.3 HD44780_create_symbol()

```
void HD44780_create_symbol (
            const HD44780 * lcd,
            uint8_t address,
            bool font_5x10,
            const uint8_t symbol[] )
```

Create a user defined character to display in the LCD. The controller memory can store up to 8 5x8 symbols, and up to 4 5x10 symbols.

Warning

5x10 symbols need 2 CGRAM slots, so after defining a 5x10 symbol at address n, the next symbol should be defined at address n+2.

Parameters

lcd	Controller instance.
address	ASCII code of the new symbol, must be in the range from 0 to 7 inclusive.
font_5x10	Whetеher the new symbol will be a 5x10 character.
symbol	Array of 5 bit values where each bit will determine whether the corresponding pixel is lit up in its corresponding row.

4.2.1.4 HD44780_cursor_to()

```
void HD44780_cursor_to (
            const HD44780 * lcd,
            uint8_t column,
            uint8_t row )
```

Move the cursor to the desired position.

Parameters

lcd	Controller instance.
column	Index of the desired cursor position in the line. Must be less than 0x50 in single line mode, and less than 0x28 in two lines mode, or the cursor will wrap to the next line causing undefined behaviour.
row	Index of the desired row. Must be 0 if the controller is configured for single line mode, and 0 or 1 when the controller is in two lines mode.

4.2.1.5 HD44780_init()

```
void HD44780_init (
            const HD44780 * lcd )
```

Initialize the necessary hardware peripherals, then configure the controller itself. The initial configuration will be the same as calling HD44780_configure() with all the config flags set to false.

Parameters

lcd	Controller instance.

4.2.1.6 HD44780_put_char()

```
void HD44780_put_char (
            const HD44780 * lcd,
            uint8_t chr )
```

Write a single character to the lcd, then advance the cursor. When the character is '\n' the cursor will advance to the next line, wrapping around from last to first. When the character is '\t' 4 spaces will be written to the display.

Parameters

lcd	Controller instance.
chr	Character to be printed to the lcd.

4.2.1.7 HD44780_put_str()

```
void HD44780_put_str (
            const HD44780 * lcd,
            const char * str )
```

Write a string to the lcd, then advance the cursor. The same considerations for special characters from HD44780_put_char() apply to this function.

Warning

The string must be null terminated.

Parameters

lcd	Controller instance.
str	String to be printed to the lcd.

4.2.1.8 HD44780_return_home()

```
void HD44780_return_home (
            const HD44780 * lcd )
```

Reset display shift to the initial position and move the cursor to position 0 of the first line.

Parameters

lcd	Controller instance.

4.2.1.9 HD44780_shift_display()

```
void HD44780_shift_display (
            const HD44780 * lcd,
            int8_t n )
```

Shift the contents of the display right or left by n positions. The first and second line will shift at the same time.

Note

The execution time of this function will increase linearly with the numbers of positions shifted (∼37us / position shifted).

Parameters

lcd	Controller instance.
n	Number of positions to shift. When the value is positive the diplay will shift left to right, when negative the shift operation will advance right to left.

4.3 HD44780.h

Go to the documentation of this file.

```
1 /*
2  * HD44780.h
3  *
4  *  Created on: Feb 2, 2022
5  *      Author: kubag
6  */
7
8 #ifndef __HD44780_H__
9 #define __HD44780_H__
10
11 #include <stdbool.h>
12 #include <stdlib.h>
13
14
15 #include "stm32f2xx_hal.h"
16
17
23 typedef struct
24 {
25     GPIO_TypeDef *rs_gpio;
26     GPIO_TypeDef *rw_gpio;
27     GPIO_TypeDef *en_gpio;
28     GPIO_TypeDef *d0_gpio;
29     GPIO_TypeDef *d1_gpio;
30     GPIO_TypeDef *d2_gpio;
31     GPIO_TypeDef *d3_gpio;
32     GPIO_TypeDef *d4_gpio;
33     GPIO_TypeDef *d5_gpio;
34     GPIO_TypeDef *d6_gpio;
35     GPIO_TypeDef *d7_gpio;
37     uint16_t rs_pin;
38     uint16_t rw_pin;
39     uint16_t en_pin;
40     uint16_t d0_pin;
41     uint16_t d1_pin;
42     uint16_t d2_pin;
43     uint16_t d3_pin;
44     uint16_t d4_pin;
45     uint16_t d5_pin;
46     uint16_t d6_pin;
47     uint16_t d7_pin;
53     bool interface_8_bit;
54
58     bool single_line;
59
65     bool font_5x10;
66 } HD44780;
67
72 typedef struct
73 {
78     bool disable_display;
79
84     bool enable_cursor;
85
90     bool enable_blink;
91
95     bool shift_display;
96
101     bool shift_rtl;
102 } HD44780_Config;
103
110 void HD44780_init(const HD44780 *lcd);
111
119 void HD44780_configure(const HD44780 *lcd, const HD44780_Config *config);
120
126 void HD44780_clear(const HD44780 *lcd);
127
133 void HD44780_return_home(const HD44780 *lcd);
134
146 void HD44780_cursor_to(const HD44780 *lcd, uint8_t column, uint8_t row);
147
160 void HD44780_shift_display(const HD44780 *lcd, int8_t n);
161
178 void HD44780_create_symbol(const HD44780 *lcd, uint8_t address, bool font_5x10, const uint8_t symbol[]);
179
189 void HD44780_put_char(const HD44780 *lcd, uint8_t chr);
190
201 void HD44780_put_str(const HD44780 *lcd, const char *str);
202
203 #endif /* __HD44780_H__ */
204
205
```

4.4 main.c File Reference

: Main program body.

```
#include "main.h"
#include "i2c.h"
#include "spi.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"
#include "bmp280_defs.h"
#include "bmp280.h"
#include "regulator.h"
#include "HD44780.h"
```

Macros

- #define BMP280_SPI (&hspi3)
- #define BMP280_CS1 1
- #define BMP280_CS2 2
- #define BMP280_SPI_BUFFER_LEN 28
- #define BMP280_DATA_INDEX 1

Functions

- int8_t bmp280_spi_reg_write (uint8_t cs, uint8_t reg_addr, uint8_t ∗reg_data, uint16_t length)

  Function for writing the sensor 's registers through SPI bus.
- int8_t bmp280_spi_reg_read (uint8_t cs, uint8_t reg_addr, uint8_t ∗reg_data, uint16_t length)

  Function for reading the sensor 's registers through SPI bus.
- void SystemClock_Config (void)

  System Clock Configuration.
- int convert (char str[ ])

  Function for converting string from external app to integer value.
- void HAL_UART_RxCpltCallback (UART_HandleTypeDef ∗huart)

  Period elapsed callback in non-blocking mode. Assigning converted data to set point value.
- void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef ∗htim)

  Period elapsed callback in non-blocking mode with implemented control algorithm.
- int main (void)

  The application entry point.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

- struct bmp280_dev bmp280_1
- char reference_value [2]
- double temp
- int set_point = 0
- uint16_t pwm_duty
- two_position_t tp = {.H = 0.5, .u_min = 0, .u_max = 999, .u_value = 0}

4.4.1 Detailed Description

: Main program body.

Attention

4.4.2 Macro Definition Documentation

4.4.2.1 BMP280_CS1

```
#define BMP280_CS1 1
```

4.4.2.2 BMP280_CS2

```
#define BMP280_CS2 2
```

4.4.2.3 BMP280_DATA_INDEX

```
#define BMP280_DATA_INDEX 1
```

4.4.2.4 BMP280_SPI

```
#define BMP280_SPI (&hspi3)
```

4.4.2.5 BMP280_SPI_BUFFER_LEN

```
#define BMP280_SPI_BUFFER_LEN 28
```

4.4.3 Function Documentation

4.4.3.1 bmp280_spi_reg_read()

```
int8_t bmp280_spi_reg_read (
            uint8_t cs,
            uint8_t reg_addr,
            uint8_t * reg_data,
            uint16_t length )
```

Function for reading the sensor 's registers through SPI bus.

Parameters

in	*cs*	: Chip select to enable the sensor .
in	*reg_addr*	: Register address .
out	*reg_data*	: Pointer to the data buffer to store the read data .
in	*length*	: No of bytes to read .

Returns

Status of execution

Return values

0	-> Success
>0	-> Failure Info

4.4.3.2 bmp280_spi_reg_write()

```
int8_t bmp280_spi_reg_write (
            uint8_t cs,
            uint8_t reg_addr,
            uint8_t * reg_data,
            uint16_t length )
```

Function for writing the sensor 's registers through SPI bus.

Parameters

in	*cs*	: Chip select to enable the sensor .
in	*reg_addr*	: Register address .
in	*reg_data*	: Pointer to the data buffer whose data has to be written
in	*length*	: No of bytes to write .

Returns

Status of execution

Return values

0	-> Success
>0	-> Failure Info

4.4.3.3 convert()

```
int convert (
             char str[] )
```

Function for converting string from external app to integer value.

Parameters

in	*str*	: String to convert.

Returns

converted string.

4.4.3.4 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.4.3.5 HAL_TIM_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback (
            TIM_HandleTypeDef * htim )
```

Period elapsed callback in non-blocking mode with implemented control algorithm.

Parameters

htim	TIM handle

Return values

None	

4.4.3.6 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
            UART_HandleTypeDef * huart )
```

Period elapsed callback in non-blocking mode. Assigning converted data to set point value.

Parameters

htim	TIM handle

Return values

None	

4.4.3.7 main()

```
int main (
            void  )
```

The application entry point.

Return values

int	

4.4.3.8 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

System Clock Configuration.

Return values

None	

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

4.4.4 Variable Documentation

4.4.4.1 bmp280_1

```
struct bmp280_dev bmp280_1
```

Initial value:

```
= {
  .dev_id = BMP280_CS1,
  .intf = BMP280_SPI_INTF,
  .read = bmp280_spi_reg_read,
  .write = bmp280_spi_reg_write,
  .delay_ms = HAL_Delay
}
```

4.4.4.2 pwm_duty

```
uint16_t pwm_duty
```

4.4.4.3 reference_value

```
char reference_value[2]
```

4.4.4.4 set_point

```
int set_point = 0
```

4.4.4.5 temp

```
double temp
```

4.4.4.6 tp

```
two_position_t tp = {.H = 0.5, .u_min = 0, .u_max = 999, .u_value = 0}
```

4.5 main.h File Reference

: Header for main.c file. This file contains the common defines of the application.

```
#include "stm32f2xx_hal.h"
```

Macros

- #define BUTTON_Pin GPIO_PIN_2
- #define BUTTON_GPIO_Port GPIOE
- #define FAN_Pin GPIO_PIN_4
- #define FAN_GPIO_Port GPIOE
- #define USER_Btn_Pin GPIO_PIN_13
- #define USER_Btn_GPIO_Port GPIOC
- #define USER_Btn_EXTI_IRQn EXTI15_10_IRQn
- #define MCO_Pin GPIO_PIN_0
- #define MCO_GPIO_Port GPIOH
- #define RMII_MDC_Pin GPIO_PIN_1
- #define RMII_MDC_GPIO_Port GPIOC
- #define RMII_REF_CLK_Pin GPIO_PIN_1
- #define RMII_REF_CLK_GPIO_Port GPIOA
- #define RMII_MDIO_Pin GPIO_PIN_2
- #define RMII_MDIO_GPIO_Port GPIOA
- #define LCD_D5_Pin GPIO_PIN_4
- #define LCD_D5_GPIO_Port GPIOA
- #define RMII_CRS_DV_Pin GPIO_PIN_7
- #define RMII_CRS_DV_GPIO_Port GPIOA
- #define RMII_RXD0_Pin GPIO_PIN_4
- #define RMII_RXD0_GPIO_Port GPIOC
- #define RMII_RXD1_Pin GPIO_PIN_5
- #define RMII_RXD1_GPIO_Port GPIOC
- #define LD1_Pin GPIO_PIN_0
- #define LD1_GPIO_Port GPIOB
- #define LCD_RW_Pin GPIO_PIN_11
- #define LCD_RW_GPIO_Port GPIOB
- #define RMII_TXD1_Pin GPIO_PIN_13
- #define RMII_TXD1_GPIO_Port GPIOB
- #define LD3_Pin GPIO_PIN_14
- #define LD3_GPIO_Port GPIOB

- #define LCD_RS_Pin GPIO_PIN_15
- #define LCD_RS_GPIO_Port GPIOB
- #define STLK_RX_Pin GPIO_PIN_8
- #define STLK_RX_GPIO_Port GPIOD
- #define STLK_TX_Pin GPIO_PIN_9
- #define STLK_TX_GPIO_Port GPIOD
- #define USB_PowerSwitchOn_Pin GPIO_PIN_6
- #define USB_PowerSwitchOn_GPIO_Port GPIOG
- #define USB_OverCurrent_Pin GPIO_PIN_7
- #define USB_OverCurrent_GPIO_Port GPIOG
- #define LCD_E_Pin GPIO_PIN_6
- #define LCD_E_GPIO_Port GPIOC
- #define LCD_D7_Pin GPIO_PIN_9
- #define LCD_D7_GPIO_Port GPIOC
- #define USB_SOF_Pin GPIO_PIN_8
- #define USB_SOF_GPIO_Port GPIOA
- #define USB_VBUS_Pin GPIO_PIN_9
- #define USB_VBUS_GPIO_Port GPIOA
- #define USB_ID_Pin GPIO_PIN_10
- #define USB_ID_GPIO_Port GPIOA
- #define USB_DM_Pin GPIO_PIN_11
- #define USB_DM_GPIO_Port GPIOA
- #define USB_DP_Pin GPIO_PIN_12
- #define USB_DP_GPIO_Port GPIOA
- #define TMS_Pin GPIO_PIN_13
- #define TMS_GPIO_Port GPIOA
- #define TCK_Pin GPIO_PIN_14
- #define TCK_GPIO_Port GPIOA
- #define SPI_CS_Pin GPIO_PIN_15
- #define SPI_CS_GPIO_Port GPIOA
- #define RMII_TX_EN_Pin GPIO_PIN_11
- #define RMII_TX_EN_GPIO_Port GPIOG
- #define RMII_TXD0_Pin GPIO_PIN_13
- #define RMII_TXD0_GPIO_Port GPIOG
- #define SWO_Pin GPIO_PIN_3
- #define SWO_GPIO_Port GPIOB
- #define LCD_D4_Pin GPIO_PIN_4
- #define LCD_D4_GPIO_Port GPIOB
- #define LCD_D6_Pin GPIO_PIN_5
- #define LCD_D6_GPIO_Port GPIOB
- #define LD2_Pin GPIO_PIN_7
- #define LD2_GPIO_Port GPIOB

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.5.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

4.5.2 Macro Definition Documentation

4.5.2.1 BUTTON_GPIO_Port

```
#define BUTTON_GPIO_Port GPIOE
```

4.5.2.2 BUTTON_Pin

```
#define BUTTON_Pin GPIO_PIN_2
```

4.5.2.3 FAN_GPIO_Port

```
#define FAN_GPIO_Port GPIOE
```

4.5.2.4 FAN_Pin

```
#define FAN_Pin GPIO_PIN_4
```

4.5.2.5 LCD_D4_GPIO_Port

```
#define LCD_D4_GPIO_Port GPIOB
```

4.5.2.6 LCD_D4_Pin

```
#define LCD_D4_Pin GPIO_PIN_4
```

4.5.2.7 LCD_D5_GPIO_Port

```
#define LCD_D5_GPIO_Port GPIOA
```

4.5.2.8 LCD_D5_Pin

```
#define LCD_D5_Pin GPIO_PIN_4
```

4.5.2.9 LCD_D6_GPIO_Port

```
#define LCD_D6_GPIO_Port GPIOB
```

4.5.2.10 LCD_D6_Pin

```
#define LCD_D6_Pin GPIO_PIN_5
```

4.5.2.11 LCD_D7_GPIO_Port

```
#define LCD_D7_GPIO_Port GPIOC
```

4.5.2.12 LCD_D7_Pin

```
#define LCD_D7_Pin GPIO_PIN_9
```

4.5.2.13 LCD_E_GPIO_Port

```
#define LCD_E_GPIO_Port GPIOC
```

4.5.2.14 LCD_E_Pin

```
#define LCD_E_Pin GPIO_PIN_6
```

4.5.2.15 LCD_RS_GPIO_Port

```
#define LCD_RS_GPIO_Port GPIOB
```

4.5.2.16 LCD_RS_Pin

```
#define LCD_RS_Pin GPIO_PIN_15
```

4.5.2.17 LCD_RW_GPIO_Port

```
#define LCD_RW_GPIO_Port GPIOB
```

4.5.2.18 LCD_RW_Pin

```
#define LCD_RW_Pin GPIO_PIN_11
```

4.5.2.19 LD1_GPIO_Port

```
#define LD1_GPIO_Port GPIOB
```

4.5.2.20 LD1_Pin

```
#define LD1_Pin GPIO_PIN_0
```

4.5.2.21 LD2_GPIO_Port

```
#define LD2_GPIO_Port GPIOB
```

4.5.2.22 LD2_Pin

```
#define LD2_Pin GPIO_PIN_7
```

4.5.2.23 LD3_GPIO_Port

```
#define LD3_GPIO_Port GPIOB
```

4.5.2.24 LD3_Pin

```
#define LD3_Pin GPIO_PIN_14
```

4.5.2.25 MCO_GPIO_Port

```
#define MCO_GPIO_Port GPIOH
```

4.5.2.26 MCO_Pin

```
#define MCO_Pin GPIO_PIN_0
```

4.5.2.27 RMII_CRS_DV_GPIO_Port

```
#define RMII_CRS_DV_GPIO_Port GPIOA
```

4.5.2.28 RMII_CRS_DV_Pin

```
#define RMII_CRS_DV_Pin GPIO_PIN_7
```

4.5.2.29 RMII_MDC_GPIO_Port

```
#define RMII_MDC_GPIO_Port GPIOC
```

4.5.2.30 RMII_MDC_Pin

```
#define RMII_MDC_Pin GPIO_PIN_1
```

4.5.2.31 RMII_MDIO_GPIO_Port

```
#define RMII_MDIO_GPIO_Port GPIOA
```

4.5.2.32 RMII_MDIO_Pin

```
#define RMII_MDIO_Pin GPIO_PIN_2
```

4.5.2.33 RMII_REF_CLK_GPIO_Port

```
#define RMII_REF_CLK_GPIO_Port GPIOA
```

4.5.2.34 RMII_REF_CLK_Pin

```
#define RMII_REF_CLK_Pin GPIO_PIN_1
```

4.5.2.35 RMII_RXD0_GPIO_Port

```
#define RMII_RXD0_GPIO_Port GPIOC
```

4.5.2.36 RMII_RXD0_Pin

```
#define RMII_RXD0_Pin GPIO_PIN_4
```

4.5.2.37 RMII_RXD1_GPIO_Port

```
#define RMII_RXD1_GPIO_Port GPIOC
```

4.5.2.38 RMII_RXD1_Pin

```
#define RMII_RXD1_Pin GPIO_PIN_5
```

4.5.2.39 RMII_TX_EN_GPIO_Port

```
#define RMII_TX_EN_GPIO_Port GPIOG
```

4.5.2.40 RMII_TX_EN_Pin

```
#define RMII_TX_EN_Pin GPIO_PIN_11
```

4.5.2.41 RMII_TXD0_GPIO_Port

```
#define RMII_TXD0_GPIO_Port GPIOG
```

4.5.2.42 RMII_TXD0_Pin

```
#define RMII_TXD0_Pin GPIO_PIN_13
```

4.5.2.43 RMII_TXD1_GPIO_Port

```
#define RMII_TXD1_GPIO_Port GPIOB
```

4.5.2.44 RMII_TXD1_Pin

```
#define RMII_TXD1_Pin GPIO_PIN_13
```

4.5.2.45 SPI_CS_GPIO_Port

```
#define SPI_CS_GPIO_Port GPIOA
```

4.5.2.46 SPI_CS_Pin

```
#define SPI_CS_Pin GPIO_PIN_15
```

4.5.2.47 STLK_RX_GPIO_Port

```
#define STLK_RX_GPIO_Port GPIOD
```

4.5.2.48 STLK_RX_Pin

```
#define STLK_RX_Pin GPIO_PIN_8
```

4.5.2.49 STLK_TX_GPIO_Port

```
#define STLK_TX_GPIO_Port GPIOD
```

4.5.2.50 STLK_TX_Pin

```
#define STLK_TX_Pin GPIO_PIN_9
```

4.5.2.51 SWO_GPIO_Port

```
#define SWO_GPIO_Port GPIOB
```

4.5.2.52 SWO_Pin

```
#define SWO_Pin GPIO_PIN_3
```

4.5.2.53 TCK_GPIO_Port

```
#define TCK_GPIO_Port GPIOA
```

4.5.2.54 TCK_Pin

```
#define TCK_Pin GPIO_PIN_14
```

4.5.2.55 TMS_GPIO_Port

```
#define TMS_GPIO_Port GPIOA
```

4.5.2.56 TMS_Pin

```
#define TMS_Pin GPIO_PIN_13
```

4.5.2.57 USB_DM_GPIO_Port

```
#define USB_DM_GPIO_Port GPIOA
```

4.5.2.58 USB_DM_Pin

```
#define USB_DM_Pin GPIO_PIN_11
```

4.5.2.59 USB_DP_GPIO_Port

```
#define USB_DP_GPIO_Port GPIOA
```

4.5.2.60 USB_DP_Pin

```
#define USB_DP_Pin GPIO_PIN_12
```

4.5.2.61 USB_ID_GPIO_Port

```
#define USB_ID_GPIO_Port GPIOA
```

4.5.2.62 USB_ID_Pin

```
#define USB_ID_Pin GPIO_PIN_10
```

4.5.2.63 USB_OverCurrent_GPIO_Port

```
#define USB_OverCurrent_GPIO_Port GPIOG
```

4.5.2.64 USB_OverCurrent_Pin

```
#define USB_OverCurrent_Pin GPIO_PIN_7
```

4.5.2.65 USB_PowerSwitchOn_GPIO_Port

```
#define USB_PowerSwitchOn_GPIO_Port GPIOG
```

4.5.2.66 USB_PowerSwitchOn_Pin

```
#define USB_PowerSwitchOn_Pin GPIO_PIN_6
```

4.5.2.67 USB_SOF_GPIO_Port

```
#define USB_SOF_GPIO_Port GPIOA
```

4.5.2.68 USB_SOF_Pin

```
#define USB_SOF_Pin GPIO_PIN_8
```

4.5.2.69 USB_VBUS_GPIO_Port

```
#define USB_VBUS_GPIO_Port GPIOA
```

4.5.2.70 USB_VBUS_Pin

```
#define USB_VBUS_Pin GPIO_PIN_9
```

4.5.2.71 USER_Btn_EXTI_IRQn

```
#define USER_Btn_EXTI_IRQn EXTI15_10_IRQn
```

4.5.2.72 USER_Btn_GPIO_Port

```
#define USER_Btn_GPIO_Port GPIOC
```

4.5.2.73 USER_Btn_Pin

```
#define USER_Btn_Pin GPIO_PIN_13
```

4.5.3 Function Documentation

4.5.3.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.6 main.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
20 /* USER CODE END Header */
21
22 /* Define to prevent recursive inclusion -------------------------------------*/
23 #ifndef __MAIN_H
```

```
24 #define __MAIN_H
25
26 #ifdef __cplusplus
27 extern "C" {
28 #endif
29
30 /* Includes ------------------------------------------------------------------*/
31 #include "stm32f2xx_hal.h"
32
33 /* Private includes ----------------------------------------------------------*/
34 /* USER CODE BEGIN Includes */
35
36 /* USER CODE END Includes */
37
38 /* Exported types ------------------------------------------------------------*/
39 /* USER CODE BEGIN ET */
40
41 /* USER CODE END ET */
42
43 /* Exported constants --------------------------------------------------------*/
44 /* USER CODE BEGIN EC */
45
46 /* USER CODE END EC */
47
48 /* Exported macro ------------------------------------------------------------*/
49 /* USER CODE BEGIN EM */
50
51 /* USER CODE END EM */
52
53 /* Exported functions prototypes ---------------------------------------------*/
54 void Error_Handler(void);
55
56 /* USER CODE BEGIN EFP */
57
58 /* USER CODE END EFP */
59
60 /* Private defines -----------------------------------------------------------*/
61 #define BUTTON_Pin GPIO_PIN_2
62 #define BUTTON_GPIO_Port GPIOE
63 #define FAN_Pin GPIO_PIN_4
64 #define FAN_GPIO_Port GPIOE
65 #define USER_Btn_Pin GPIO_PIN_13
66 #define USER_Btn_GPIO_Port GPIOC
67 #define USER_Btn_EXTI_IRQn EXTI15_10_IRQn
68 #define MCO_Pin GPIO_PIN_0
69 #define MCO_GPIO_Port GPIOH
70 #define RMII_MDC_Pin GPIO_PIN_1
71 #define RMII_MDC_GPIO_Port GPIOC
72 #define RMII_REF_CLK_Pin GPIO_PIN_1
73 #define RMII_REF_CLK_GPIO_Port GPIOA
74 #define RMII_MDIO_Pin GPIO_PIN_2
75 #define RMII_MDIO_GPIO_Port GPIOA
76 #define LCD_D5_Pin GPIO_PIN_4
77 #define LCD_D5_GPIO_Port GPIOA
78 #define RMII_CRS_DV_Pin GPIO_PIN_7
79 #define RMII_CRS_DV_GPIO_Port GPIOA
80 #define RMII_RXD0_Pin GPIO_PIN_4
81 #define RMII_RXD0_GPIO_Port GPIOC
82 #define RMII_RXD1_Pin GPIO_PIN_5
83 #define RMII_RXD1_GPIO_Port GPIOC
84 #define LD1_Pin GPIO_PIN_0
85 #define LD1_GPIO_Port GPIOB
86 #define LCD_RW_Pin GPIO_PIN_11
87 #define LCD_RW_GPIO_Port GPIOB
88 #define RMII_TXD1_Pin GPIO_PIN_13
89 #define RMII_TXD1_GPIO_Port GPIOB
90 #define LD3_Pin GPIO_PIN_14
91 #define LD3_GPIO_Port GPIOB
92 #define LCD_RS_Pin GPIO_PIN_15
93 #define LCD_RS_GPIO_Port GPIOB
94 #define STLK_RX_Pin GPIO_PIN_8
95 #define STLK_RX_GPIO_Port GPIOD
96 #define STLK_TX_Pin GPIO_PIN_9
97 #define STLK_TX_GPIO_Port GPIOD
98 #define USB_PowerSwitchOn_Pin GPIO_PIN_6
99 #define USB_PowerSwitchOn_GPIO_Port GPIOG
100 #define USB_OverCurrent_Pin GPIO_PIN_7
101 #define USB_OverCurrent_GPIO_Port GPIOG
102 #define LCD_E_Pin GPIO_PIN_6
103 #define LCD_E_GPIO_Port GPIOC
104 #define LCD_D7_Pin GPIO_PIN_9
105 #define LCD_D7_GPIO_Port GPIOC
106 #define USB_SOF_Pin GPIO_PIN_8
107 #define USB_SOF_GPIO_Port GPIOA
108 #define USB_VBUS_Pin GPIO_PIN_9
109 #define USB_VBUS_GPIO_Port GPIOA
110 #define USB_ID_Pin GPIO_PIN_10
```

```
111 #define USB_ID_GPIO_Port GPIOA
112 #define USB_DM_Pin GPIO_PIN_11
113 #define USB_DM_GPIO_Port GPIOA
114 #define USB_DP_Pin GPIO_PIN_12
115 #define USB_DP_GPIO_Port GPIOA
116 #define TMS_Pin GPIO_PIN_13
117 #define TMS_GPIO_Port GPIOA
118 #define TCK_Pin GPIO_PIN_14
119 #define TCK_GPIO_Port GPIOA
120 #define SPI_CS_Pin GPIO_PIN_15
121 #define SPI_CS_GPIO_Port GPIOA
122 #define RMII_TX_EN_Pin GPIO_PIN_11
123 #define RMII_TX_EN_GPIO_Port GPIOG
124 #define RMII_TXD0_Pin GPIO_PIN_13
125 #define RMII_TXD0_GPIO_Port GPIOG
126 #define SWO_Pin GPIO_PIN_3
127 #define SWO_GPIO_Port GPIOB
128 #define LCD_D4_Pin GPIO_PIN_4
129 #define LCD_D4_GPIO_Port GPIOB
130 #define LCD_D6_Pin GPIO_PIN_5
131 #define LCD_D6_GPIO_Port GPIOB
132 #define LD2_Pin GPIO_PIN_7
133 #define LD2_GPIO_Port GPIOB
134 /* USER CODE BEGIN Private defines */
135
136 /* USER CODE END Private defines */
137
138 #ifdef __cplusplus
139 }
140 #endif
141
142 #endif /* __MAIN_H */
```

4.7 regulator.c File Reference

: This file provides code for two position controller.

```
#include "regulator.h"
```

Functions

- float32_t calculate_two_position_controller (two_position_t *controller, float32_t setpoint, float32_t measured)

  Calculation of two-position controller.

4.7.1 Detailed Description

: This file provides code for two position controller.

Attention

4.7.2 Function Documentation

4.7.2.1 calculate_two_position_controller()

```
float32_t calculate_two_position_controller (
            two_position_t * controller,
            float32_t setpoint,
            float32_t measured )
```

Calculation of two-position controller.

Parameters

in,out	*controller*	: A pointer to two_position controller parameters and history.
in	*setpoint*	: Reference value.
in	*measured*	: Measured value. return controller output value

4.8 regulator.h File Reference

: Header for regulator.c file. This file contains the function prototype for regulator.c file

Data Structures

- struct two_position_t

Typedefs

- typedef float float32_t

Functions

Controller configuration structure

- float32_t calculate_two_position_controller (two_position_t ∗controller, float32_t setpoint, float32_t measured)
  Calculation of two-position controller.

4.8.1 Detailed Description

: Header for regulator.c file. This file contains the function prototype for regulator.c file

Attention

4.8.2 Typedef Documentation

4.8.2.1 float32_t

```
typedef float float32_t
```

4.8.3 Function Documentation

4.8.3.1 calculate_two_position_controller()

```
float32_t calculate_two_position_controller (
            two_position_t * controller,
            float32_t setpoint,
            float32_t measured )
```

Calculation of two-position controller.

Parameters

in,out	*controller*	: A pointer to two_position controller parameters and history.
in	*setpoint*	: Reference value.
in	*measured*	: Measured value. return controller output value

4.9 regulator.h

Go to the documentation of this file.

```
1
20 #ifndef INC_REGULATOR_H_
21 #define INC_REGULATOR_H_
22
23 typedef float float32_t;
24
26 typedef struct{
27   float32_t H;
```

```
28   float32_t u_min, u_max;
29   float32_t u_value;
30 }two_position_t;
31
32
40 float32_t calculate_two_position_controller(two_position_t* controller, float32_t setpoint, float32_t
       measured);
41
42 #endif /* INC_REGULATOR_H_ */
```

Index