My Project

Generated by Doxygen 1.9.8

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 ble_config_t Struct Reference

Main BLE configuration struct. Contains gatts and gap configuration.

`#include <ble_api.h>`

Public Attributes

- ble_gatts_t * **gatt_config**
- ble_gap_t * **gap_config**
- esp_bt_controller_config_t **bt_cfg**

3.1.1 Detailed Description

Main BLE configuration struct. Contains gatts and gap configuration.

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/ble/ble_api.h

3.2 ble_gap_t Struct Reference

Public Attributes

- ble_gap_conf_type_t **conf_type**
- esp_ble_adv_params_t **adv_params**
- esp_ble_adv_data_t **adv_data**
- esp_ble_adv_data_t **scan_rsp_data**
- ble_gap_event_handler **event_handler_cb**

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/ble/ble_gap_conf.h

3.3 ble_gatts_profile_t Struct Reference

BLE GATT configuration structure.

`#include <ble_gatt_conf.h>`

Public Attributes

- esp_gatts_cb_t **gatts_cb**
- uint16_t **gatts_if**
- uint16_t **conn_id**
- uint16_t **service_handle**
- esp_gatt_srvc_id_t **service_id**
- uint16_t **char_handle**
- esp_bt_uuid_t **char_uuid**
- esp_gatt_perm_t **perm**
- esp_gatt_char_prop_t **property**
- uint16_t **descr_handle**
- esp_bt_uuid_t **descr_uuid**
- esp_gatts_attr_db_t **gatt_db** [GATTS_ATTRIBUTES_DB_MAX_NUM]

3.3.1 Detailed Description

BLE GATT configuration structure.

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/ble/ble_gatt_conf.h

3.4 ble_gatts_t Struct Reference

Main gatts configuration structure.

`#include <ble_gatt_conf.h>`

Public Attributes

- uint16_t **profiles_num**
- ble_gatts_event_handler **event_handler_cb**
- ble_gatts_profile_t **profiles** [GATTS_PROFILES_NUM_MAX]

3.4.1 Detailed Description

Main gatts configuration structure.

Parameters

in	*profiles_num*	Number of profiles
in	*profiles*	Array of profiles
in	*event_handler_cb*	Event handler callback

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/ble/ble_gatt_conf.h

3.5 led_driver_t Struct Reference

Intended for LEDs that are positive voltage driven!

`#include <led_driver.h>`

Public Attributes

- ledc_mode_t **ledc_mode**
- uint8_t **led_gpio_num**
- uint8_t **ledc_channel_num**
- uint8_t **ledc_timer_num**
- uint16_t **duty**
- uint16_t **max_duty**
- led_state_t **toggle**

3.5.1 Detailed Description

Intended for LEDs that are positive voltage driven!

Parameters

led_gpio_num	GPIO number of LED
ledc_channel_num	LEDC channel number
ledc_timer_num	LEDC timer number
duty	Duty cycle in range
max_duty	Maximum duty cycle
toggle	Toggle LED on/off

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/misc/led_driver.h

3.6 lora_struct_t Struct Reference

Public Attributes

- lora_SPI_transmit **_spi_transmit**
- lora_delay **_delay**
- lora_GPIO_set_level **_gpio_set_level**
- lora_log **log**
- uint8_t **rst_gpio_num**
- uint8_t **cs_gpio_num**
- uint8_t **d0_gpio_num**
- int16_t **implicit_header**
- int32_t **frequency**

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/lora/lora.h

3.7 mcu_i2c_config_t Struct Reference

Public Attributes

- i2c_port_t **port**
- i2c_cmd_handle_t * **cmd**
- gpio_num_t **sda**
- gpio_num_t **scl**
- uint32_t **clk_speed**
- uint32_t **timeout**
- bool **i2c_init_flag**

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/mcu_config/mcu_i2c_↩ config.h

3.8 mcu_spi_config_t Struct Reference

Public Attributes

- spi_device_handle_t **spi**
- spi_bus_config_t **bus_config**
- spi_device_interface_config_t **dev_config**
- bool **spi_init_flag**

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/mcu_config/mcu_spi_↩ config.h

3.9 out_column_t Union Reference

Public Attributes

- uint32_t **u32**
- uint8_t **u8** [4]

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/ssd1306/ssd1306.c

3.10 PAGE_t Struct Reference

SSD1306 display page structure.

`#include <ssd1306.h>`

Public Attributes

- uint8_t **segments** [OLED_BUFFER_SIZE]

3.10.1 Detailed Description

SSD1306 display page structure.

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/ssd1306/ssd1306.h

3.11 prepare_type_env_t Struct Reference

Public Attributes

- uint8_t * **prepare_buf**
- int **prepare_len**

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/ble/test/test_ble_api.c

3.12 rgb_led_driver_t Struct Reference

RGB LED driver struct for ESP32.

`#include <rgb_led_driver.h>`

Public Attributes

- led_driver_t **led_drv** [MAX_COLOR_INDEX]
- uint16_t **max_duty**

3.12.1 Detailed Description

RGB LED driver struct for ESP32.

Parameters

led_drv	LED driver struct
max_duty	Maximum duty cycle:

Note

forces all colors to have the same

Parameters

inverted	Inverted LED logic: for common anode set to true, for common cathode set to false

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/misc/rgb_led_driver.h

3.13 ROSALIA_devices_t Struct Reference

Public Attributes

- mcu_spi_config_t **spi**
- mcu_i2c_config_t **i2c**
- mcu_twai_config_t **twai**
- lora_struct_t **lora**
- ssd1306_t **oled_display**

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/app/devices_config.h

3.14 ssd1306_t Struct Reference

SSD1306 display structure.

```
#include <ssd1306.h>
```

Public Attributes

- ssd1306_i2c_master_write_byte **_i2c_master_write_byte**
- ssd1306_i2c_master_write **_i2c_master_write**
- ssd1306_i2c_master_start **_i2c_master_start**
- ssd1306_i2c_master_stop **_i2c_master_stop**
- ssd1306_i2c_master_cmd_begin **_i2c_master_cmd_begin**
- ssd1306_i2c_cmd_link_create **_i2c_cmd_link_create**
- ssd1306_i2c_cmd_link_delete **_i2c_cmd_link_delete**
- ssd1306_delay **_delay**
- ssd1306_log **_log**
- uint8_t **i2c_master_write_flag**
- uint8_t **width**
- uint8_t **height**
- int **pages**
- bool **scroll_enable**
- int **scroll_start**
- int **scroll_end**
- int **scroll_direction**
- PAGE_t **screen_pages** [8]
- bool **flip**
- uint8_t **i2c_address**

3.14.1 Detailed Description

SSD1306 display structure.

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/ssd1306/ssd1306.h

3.15 twai_config_t Struct Reference

TWAI configuration structure.

`#include <mcu_twai_config.h>`

Public Attributes

- gpio_num_t **tx_gpio_num**
- gpio_num_t **rx_gpio_num**
- twai_mode_t **mode**

3.15.1 Detailed Description

TWAI configuration structure.

Parameters

tx_gpio_num	GPIO number for TX pin
rx_gpio_num	GPIO number for RX pin
mode	TWAI mode
bitrate	TWAI bitrate

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/mcu_config/mcu_twai_config.h

3.16 voltage_measure_config_t Struct Reference

Voltage measure struct.

`#include <mcu_adc_config.h>`

Public Attributes

- float **adc_cal** [MAX_ADC_CHANNELS]
- uint8_t **adc_chan** [MAX_ADC_CHANNELS]
- uint8_t **adc_chan_num**
- adc_oneshot_unit_init_cfg_t **oneshot_unit_cfg**
- adc_oneshot_chan_cfg_t **oneshot_chan_cfg**
- adc_oneshot_unit_handle_t * **oneshot_unit_handle**

3.16.1 Detailed Description

Voltage measure struct.

Parameters

adc_cal	- calibration value to be configured. voltage = rawRead * adc_cal
adc_chan	- specific channel of ADC
adc_chan_num	- number of channels to be configured
oneshot_unit_cfg	- config for ADC channel
oneshot_unit_handle	- handle for ADC channel
oneshot_chan_cfg	- config for ADC channel
oneshot_chan_handle	- handle for ADC channel

Note

adc_cal and adc_chan are arrays of size adc_chan_num. adc_cal[i] is calibration value for adc_chan[i] adc↩ _chan[i] is channel number for adc_cal[i]

oneshot_unit_cfg and oneshot_chan_cfg are the same for all channels.

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

- /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/mcu_config/mcu_adc_↩ config.h

Chapter 4

File Documentation

4.1 ble.h

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #include "ble_api.h"
00004 #include "ble_gap_conf.h"
00005 #include "ble_gatt_conf.h"
00006 #include "freertos/FreeRTOS.h"
00007 #include "freertos/task.h"
00008
00009 #define BLE_DEVICE_NAME "ROSALIA"
00010
00011 void gap_event_handler(esp_gap_ble_cb_event_t event,
00012                        esp_ble_gap_cb_param_t* param);
00013
00014 void gatt_profile_a_event_handler(esp_gatts_cb_event_t event,
00015                                   esp_gatt_if_t gatts_if,
00016                                   esp_ble_gatts_cb_param_t* param);
00017
00018 void gatts_event_handler(esp_gatts_cb_event_t event, esp_gatt_if_t gatts_if,
00019                          esp_ble_gatts_cb_param_t* param);
00020
00021 bool ble_config_init(void);
00022
00023 void ble_init_task(void* arg);
```

4.2 devices_config.h

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #pragma once
00004
00005 #include "mcu_i2c_config.h"
00006 #include "lora.h "
00007 #include "lora_esp32_config.h"
00008 #include "mcu_spi_config.h"
00009 #include "ssd1306_esp32_config.h"
00010 #include "mcu_uart_config.h"
00011 #include "mcu_twai_config.h"
00012
00013 typedef struct {
00014   mcu_spi_config_t spi;
00015   mcu_i2c_config_t i2c;
00016   mcu_twai_config_t twai;
00017   lora_struct_t lora;
00018   ssd1306_t oled_display;
00019 } ROSALIA_devices_t;
```

4.3 slave_communication.h

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #pragma once
00004
00005 #include "devices_config.h"
```

4.4 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/app/user_interface.h File Reference

User interface app header file Contains task for rgb LED, status LED, buzzer and button handling.

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "mcu_adc_config.h"
#include "rgb_led_driver.h"
#include "sdkconfig.h"
#include "ssd1306_esp32_config.h"
```

Macros

- #define **LEDC_FREQ_HZ** 5000
- #define **LEDC_DUTY_RES** LEDC_TIMER_13_BIT
- #define **LEDC_HS_TIMER** LEDC_TIMER_0
- #define **LEDC_HS_MODE** LEDC_LOW_SPEED_MODE
- #define **MAX_DUTY** 8192

Enumerations

- enum rgb_led_color_t {
  NONE = 0b000 , **RED** = 0b001 , **GREEN** = 0b010 , **BLUE** = 0b100 ,
  YELLOW = RED | GREEN , **CYAN** = GREEN | BLUE , **MAGENTA** = RED | BLUE , **WHITE** = RED | GREEN | BLUE }

  RGB LED color enum.
- enum **adc_chan_cfg_t** { **CAN_CHANNEL** = ADC_CHANNEL_0 , **VBAT_CHANNEL** = ADC_CHANNEL_1 , **ADJV_CHANNEL** = ADC_CHANNEL_3 }
- enum **adc_chan_index_cfg_t** { **CAN_CHANNEL_INDEX** = 0 , **VBAT_CHANNEL_INDEX** , **ADJV_↩ CHANNEL_INDEX** , **MAX_CHANNEL_INDEX** }

Functions

- void **user_interface_task** (void ∗arg)
- void **init_user_interface_task** (void ∗arg)

4.4.1 Detailed Description

User interface app header file Contains task for rgb LED, status LED, buzzer and button handling.

4.5 user_interface.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002 #pragma once
00003
00004 #include "freertos/FreeRTOS.h"
00005 #include "freertos/task.h"
00006 #include "mcu_adc_config.h"
00007 #include "rgb_led_driver.h"
00008 #include "sdkconfig.h"
00009 #include "ssd1306_esp32_config.h"
00010
00017 #define LEDC_FREQ_HZ 5000
00018 #define LEDC_DUTY_RES LEDC_TIMER_13_BIT
00019 #define LEDC_HS_TIMER LEDC_TIMER_0
00020 #define LEDC_HS_MODE LEDC_LOW_SPEED_MODE
00021 #define MAX_DUTY 8192  // 2**13
00022
00026 typedef enum {
00027   NONE = 0b000,
00028   RED = 0b001,
00029   GREEN = 0b010,
00030   BLUE = 0b100,
00031   YELLOW = RED | GREEN,
00032   CYAN = GREEN | BLUE,
00033   MAGENTA = RED | BLUE,
00034   WHITE = RED | GREEN | BLUE
00035 } rgb_led_color_t;
00036
00037 typedef enum {
00038   CAN_CHANNEL = ADC_CHANNEL_0,
00039   VBAT_CHANNEL = ADC_CHANNEL_1,
00040   ADJV_CHANNEL = ADC_CHANNEL_3,
00041 } adc_chan_cfg_t;
00042
00043 typedef enum {
00044   CAN_CHANNEL_INDEX = 0,
00045   VBAT_CHANNEL_INDEX,
00046   ADJV_CHANNEL_INDEX,
00047   MAX_CHANNEL_INDEX
00048 } adc_chan_index_cfg_t;
00049
00050 void user_interface_task(void* arg);
00051
00052 void init_user_interface_task(void* arg);
```

4.6 lora_test_config.h

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #include "config.h"
00004 #include "driver/uart.h"
00005 #include "esp_console.h"
00006 #include "esp_log.h"
00007 #include "esp_system.h"
00008 #include "esp_vfs_dev.h"
00009 #include "esp_vfs_fat.h"
00010 #include "freertos/FreeRTOS.h"
00011 #include "lora_esp32_config.h"
00012 #include "lora.h"
00013 #include "sdkconfig.h"
00014
00015 #define TAG "MAIN"
00016 // TODO(Glibus): Change the GPIO nums to another struct apply in
00017 // lora_esp32_config files
00018 lora_struct_t lora = {._spi_transmit = _lora_SPI_transmit,
00019                       ._delay = _lora_delay,
00020                       ._gpio_set_level = _lora_GPIO_set_level,
00021                       .log = _lora_log,
00022                       .rst_gpio_num = CONFIG_LORA_RS,
00023                       .cs_gpio_num = CONFIG_LORA_CS,
00024                       .d0_gpio_num = CONFIG_LORA_D0,
00025                       .implicit_header = 0,
00026                       .frequency = 0};
00027
00031 void lora_test_init();
00032
00037 void task_lora_tx(void *param);
00038
00043 void task_lora_rx(void *param);
```

4.7 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/ble/ble_api.h File Reference

BLE API - contains all BLE functions and structures ass well as attributes of GATT - unified in one structure.

```
#include <stdint.h>
#include "ble_gap_conf.h"
#include "ble_gatt_conf.h"
#include "esp_bt.h"
#include "esp_bt_defs.h"
#include "esp_bt_main.h"
#include "nvs_flash.h"
#include "sdkconfig.h"
```

Classes

- struct ble_config_t

  Main BLE configuration struct. Contains gatts and gap configuration.

Macros

- #define BLE_UUID_CONFIG_DEFAULT()
- #define BLE_ADV_DATA_CONFIG_DEFAULT()
- #define BLE_SCAN_RSP_DATA_CONFIG_DEFAULT()
- #define BLE_ADV_PARAMS_CONFIG_DEFAULT()

Enumerations

- enum ble_err_t { **BLE_OK** = 0 , **BLE_HARDWARE_INIT_ERR** , **BLE_ERR** }

  BLE status enum.

Functions

- ble_err_t **ble_esp_hardware_init** (ble_config_t *ble)

  Initiates all necessary esp hardware related to bluetooth.
- ble_err_t **ble_init** (ble_config_t *ble)

  Initiates the whole bluetooth stack, as well as gap ble_gatts_t and ble_gap_t configuration.
- const char * ble_err_to_string (ble_err_t err)

  Converts ble_err_t to string.

4.7.1 Detailed Description

BLE API - contains all BLE functions and structures ass well as attributes of GATT - unified in one structure.

4.7.2 Macro Definition Documentation

4.7.2.1 BLE_ADV_DATA_CONFIG_DEFAULT

```
#define BLE_ADV_DATA_CONFIG_DEFAULT( )
```

Value:

```
  {                                                                       \
    .set_scan_rsp = false, .include_name = true, .include_txpower = false,     \
    .min_interval = 0x0006, .max_interval = 0x0010, .appearance = 0x00,        \
    .manufacturer_len = 0, .p_manufacturer_data = NULL, .service_data_len = 0, \
    .p_service_data = NULL, .service_uuid_len = sizeof(adv_service_uuid128),   \
    .p_service_uuid = adv_service_uuid128,                                     \
    .flag = (ESP_BLE_ADV_FLAG_GEN_DISC | ESP_BLE_ADV_FLAG_BREDR_NOT_SPT),     \
  }
```

4.7.2.2 BLE_ADV_PARAMS_CONFIG_DEFAULT

```
#define BLE_ADV_PARAMS_CONFIG_DEFAULT( )
```

Value:

```
  {                                                                  \
    .adv_int_min = 0x20, .adv_int_max = 0x40, .adv_type = ADV_TYPE_IND, \
    .own_addr_type = BLE_ADDR_TYPE_PUBLIC, .channel_map = ADV_CHNL_ALL, \
    .adv_filter_policy = ADV_FILTER_ALLOW_SCAN_ANY_CON_ANY,            \
  }
```

4.7.2.3 BLE_SCAN_RSP_DATA_CONFIG_DEFAULT

```
#define BLE_SCAN_RSP_DATA_CONFIG_DEFAULT( )
```

Value:

```
  {                                                                    \
    .set_scan_rsp = true, .include_name = true, .include_txpower = true,    \
    .appearance = 0x00, .manufacturer_len = 0, .p_manufacturer_data = NULL, \
    .service_data_len = 0, .p_service_data = NULL,                       \
    .service_uuid_len = sizeof(adv_service_uuid128),                     \
    .p_service_uuid = adv_service_uuid128,                               \
    .flag = (ESP_BLE_ADV_FLAG_GEN_DISC | ESP_BLE_ADV_FLAG_BREDR_NOT_SPT),  \
  }
```

4.7.2.4 BLE_UUID_CONFIG_DEFAULT

```
#define BLE_UUID_CONFIG_DEFAULT( )
```

Value:

```
  {                                                                    \
    0xfb, 0x34, 0x9b, 0x5f, 0x80, 0x00, 0x00, 0x80, 0x00, 0x10, 0x00, 0x00, \
        0xEE, 0x00, 0x00, 0x00, 0xfb, 0x34, 0x9b, 0x5f, 0x80, 0x00, 0x00,   \
        0x80, 0x00, 0x10, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00,               \
  }
```

4.7.3 Function Documentation

4.7.3.1 ble_err_to_string()

```
const char * ble_err_to_string (
            ble_err_t err )
```

Converts ble_err_t to string.

Parameters

in	*err*	- error to convert

Returns

string representation of error

4.8 ble_api.h

Go to the documentation of this file.

```
// Copyright 2023 PWr in Space, Krzysztof Gliwiński
#pragma once

#include <stdint.h>

#include "ble_gap_conf.h"
#include "ble_gatt_conf.h"
#include "esp_bt.h"
#include "esp_bt_defs.h"
#include "esp_bt_main.h"
#include "nvs_flash.h"
#include "sdkconfig.h"

/* LSB
 <-------------------------------------------------------------------------------->
 MSB */
// first uuid, 16bit, [12],[13] is the value
// second uuid, 32bit, [12], [13], [14], [15] is the value
#define BLE_UUID_CONFIG_DEFAULT()                                                   \
  {                                                                                 \
    0xfb, 0x34, 0x9b, 0x5f, 0x80, 0x00, 0x00, 0x80, 0x00, 0x10, 0x00, 0x00, \
        0xEE, 0x00, 0x00, 0x00, 0xfb, 0x34, 0x9b, 0x5f, 0x80, 0x00, 0x00,   \
        0x80, 0x00, 0x10, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00,               \
  }

// slave connection min interval, Time = min_interval * 1.25 msec
// slave connection max interval, Time = max_interval * 1.25 msec
// TEST_MANUFACTURER_DATA_LEN,
// &test_manufacturer[0],
#define BLE_ADV_DATA_CONFIG_DEFAULT()                                              \
  {                                                                                \
    .set_scan_rsp = false, .include_name = true, .include_txpower = false,        \
    .min_interval = 0x0006, .max_interval = 0x0010, .appearance = 0x00,           \
    .manufacturer_len = 0, .p_manufacturer_data = NULL, .service_data_len = 0,    \
    .p_service_data = NULL, .service_uuid_len = sizeof(adv_service_uuid128),      \
    .p_service_uuid = adv_service_uuid128,                                         \
    .flag = (ESP_BLE_ADV_FLAG_GEN_DISC | ESP_BLE_ADV_FLAG_BREDR_NOT_SPT),         \
  }

// TEST_MANUFACTURER_DATA_LEN,
// &test_manufacturer[0],
#define BLE_SCAN_RSP_DATA_CONFIG_DEFAULT()                                        \
  {                                                                               \
    .set_scan_rsp = true, .include_name = true, .include_txpower = true,         \
    .appearance = 0x00, .manufacturer_len = 0, .p_manufacturer_data = NULL,      \
    .service_data_len = 0, .p_service_data = NULL,                               \
    .service_uuid_len = sizeof(adv_service_uuid128),                              \
    .p_service_uuid = adv_service_uuid128,                                        \
    .flag = (ESP_BLE_ADV_FLAG_GEN_DISC | ESP_BLE_ADV_FLAG_BREDR_NOT_SPT),        \
  }

#define BLE_ADV_PARAMS_CONFIG_DEFAULT()                                       \
  {                                                                           \
    .adv_int_min = 0x20, .adv_int_max = 0x40, .adv_type = ADV_TYPE_IND,      \
    .own_addr_type = BLE_ADDR_TYPE_PUBLIC, .channel_map = ADV_CHNL_ALL,      \
    .adv_filter_policy = ADV_FILTER_ALLOW_SCAN_ANY_CON_ANY,                   \
  }

typedef enum { BLE_OK = 0, BLE_HARDWARE_INIT_ERR, BLE_ERR } ble_err_t;

typedef struct {
  ble_gatts_t *gatt_config;
  ble_gap_t *gap_config;
  esp_bt_controller_config_t bt_cfg;
} ble_config_t;

```

```
00083 ble_err_t ble_esp_hardware_init(ble_config_t *ble);
00084
00089 ble_err_t ble_init(ble_config_t *ble);
00090
00096 const char *ble_err_to_string(ble_err_t err);
```

4.9 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/ble/ble_gap_conf.h File Reference

API for configurating the BLE GAP module.

```
#include <stdint.h>
#include "esp_gap_ble_api.h"
#include "esp_log.h"
```

Classes

- struct ble_gap_t

Typedefs

- typedef void(∗ ble_gap_event_handler) (esp_gap_ble_cb_event_t event, esp_ble_gap_cb_param_t ∗param)
  Handle GAP events.

Enumerations

- enum ble_gap_conf_type_t { **BLE_GAP_BROADCASTER** = 0x01b , **BLE_GAP_CENTRAL** = 0x10b , **BLE↩ _GAP_BROADCASTER_CENTRAL** = 0x11b }
  GAP configuration structure Used as a central device, only broadcaster, peripheral or both can be set.

Functions

- bool ble_gap_init (ble_gap_t ∗gap_conf)
  Initialize the BLE GAP module.

4.9.1 Detailed Description

API for configurating the BLE GAP module.

4.9.2 Typedef Documentation

4.9.2.1 ble_gap_event_handler

```
typedef void(* ble_gap_event_handler) (esp_gap_ble_cb_event_t event, esp_ble_gap_cb_param_↩
t *param)
```

Handle GAP events.

Parameters

in	*event*	GAP event
in	*param*	GAP event parameters

4.9.3 Function Documentation

4.9.3.1 ble_gap_init()

```
bool ble_gap_init (
            ble_gap_t * gap_conf )
```

Initialize the BLE GAP module.

Parameters

in	*gap_conf*	GAP configuration structure

4.10 ble_gap_conf.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002 #pragma once
00003
00004 #include <stdint.h>
00005
00006 #include "esp_gap_ble_api.h"
00007 #include "esp_log.h"
00008
00018 typedef enum {
00019   BLE_GAP_BROADCASTER = 0x01b,
00020   BLE_GAP_CENTRAL = 0x10b,
00021   BLE_GAP_BROADCASTER_CENTRAL = 0x11b
00022 } ble_gap_conf_type_t;
00023
00029 typedef void (*ble_gap_event_handler)(esp_gap_ble_cb_event_t event,
00030                                       esp_ble_gap_cb_param_t *param);
00031
00032 typedef struct {
00033   ble_gap_conf_type_t conf_type;
00034   esp_ble_adv_params_t adv_params;
00035   esp_ble_adv_data_t adv_data;
00036   esp_ble_adv_data_t scan_rsp_data;
00037   ble_gap_event_handler event_handler_cb;
00038 } ble_gap_t;
00039
00044 bool ble_gap_init(ble_gap_t *gap_conf);
```

4.11 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/ble/ble_gatt_conf.h File Reference

BLE GATT configuration structure.

```
#include <stdint.h>
#include "esp_err.h"
#include "esp_gatt_common_api.h"
```

```
#include "esp_gatt_defs.h"
#include "esp_gatts_api.h"
#include "esp_log.h"
```

Classes

- struct ble_gatts_profile_t

  BLE GATT configuration structure.
- struct ble_gatts_t

  Main gatts configuration structure.

Macros

- #define **GATTS_PROFILES_NUM_MAX** 16

  Maximum number of GATT profiles.
- #define **GATTS_ATTRIBUTES_DB_MAX_NUM** 256

  Maximum number of GATT attributes in db.

Typedefs

- typedef void(∗ ble_gatts_event_handler) (esp_gatts_cb_event_t event, esp_gatt_if_t gatts_if, esp_ble_↩ gatts_cb_param_t ∗param)

  Handle GATT events.

Functions

- esp_err_t ble_gatt_init (ble_gatts_t ∗gatts_conf)

  Initialize the BLE GATT module.
- esp_err_t ble_gatt_register_event (ble_gatts_t ∗gatts_conf)

  Register the BLE GATT module.

4.11.1 Detailed Description

BLE GATT configuration structure.

4.11.2 Typedef Documentation

4.11.2.1 ble_gatts_event_handler

```
typedef void(* ble_gatts_event_handler) (esp_gatts_cb_event_t event, esp_gatt_if_t gatts_if,
esp_ble_gatts_cb_param_t *param)
```

Handle GATT events.

Parameters

in	*event*	GATT event
in	*param*	GATT event parameters

4.11.3 Function Documentation

4.11.3.1 ble_gatt_init()

```
esp_err_t ble_gatt_init (
            ble_gatts_t * gatts_conf )
```

Initialize the BLE GATT module.

Parameters

in	*gatts_conf*	GATT configuration structure

4.11.3.2 ble_gatt_register_event()

```
esp_err_t ble_gatt_register_event (
            ble_gatts_t * gatts_conf )
```

Register the BLE GATT module.

Parameters

in	*gatts_conf*	GATT configuration structure

4.12 ble_gatt_conf.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002 #pragma once
00003
00004 #include <stdint.h>
00005
00006 #include "esp_err.h"
00007 #include "esp_gatt_common_api.h"
00008 #include "esp_gatt_defs.h"
00009 #include "esp_gatts_api.h"
00010 #include "esp_log.h"
00011
00020 #define GATTS_PROFILES_NUM_MAX 16
00021
00025 #define GATTS_ATTRIBUTES_DB_MAX_NUM 256
00026
00032 typedef void (*ble_gatts_event_handler)(esp_gatts_cb_event_t event,
00033                                         esp_gatt_if_t gatts_if,
00034                                         esp_ble_gatts_cb_param_t *param);
00035
00039 typedef struct {
00040   //   esp_gatts_attr_db_t *gatt_db;
00041   esp_gatts_cb_t gatts_cb;
```

```
00042   uint16_t gatts_if;
00043   uint16_t conn_id;
00044   uint16_t service_handle;
00045   esp_gatt_srvc_id_t service_id;
00046   uint16_t char_handle;
00047   esp_bt_uuid_t char_uuid;
00048   esp_gatt_perm_t perm;
00049   esp_gatt_char_prop_t property;
00050   uint16_t descr_handle;
00051   esp_bt_uuid_t descr_uuid;
00052   esp_gatts_attr_db_t gatt_db[GATTS_ATTRIBUTES_DB_MAX_NUM];
00053 } ble_gatts_profile_t;
00054
00061 typedef struct {
00062   uint16_t profiles_num;
00063   ble_gatts_event_handler event_handler_cb;
00064   ble_gatts_profile_t profiles[GATTS_PROFILES_NUM_MAX];
00065 } ble_gatts_t;
00066
00071 esp_err_t ble_gatt_init(ble_gatts_t *gatts_conf);
00072
00077 esp_err_t ble_gatt_register_event(ble_gatts_t *gatts_conf);
```

4.13 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/lora/lora.c File Reference

RFM95w LoRa library - multi-MCU.

```
#include "lora.h"
```

Macros

- #define **TAG** "LORA"

Functions

- lora_err_t **lora_init** (lora_struct_t *lora)

  Perform hardware initialization.
- lora_err_t **lora_default_config** (lora_struct_t *lora)

  Create default config and set parameters.
- lora_err_t lora_write_reg (lora_struct_t *lora, int16_t reg, int16_t val)

  Write a value to a register.
- uint8_t lora_read_reg (lora_struct_t *lora, int16_t reg)

  Read the current value of a register.
- void lora_reset (lora_struct_t *lora)

  Perform physical reset on the Lora chip.
- lora_err_t **lora_explicit_header_mode** (lora_struct_t *lora)

  Configure explicit header mode. Packet size will be included in the frame.
- lora_err_t lora_implicit_header_mode (lora_struct_t *lora, int16_t size)

  Configure implicit header mode. All packets will have a predefined size.
- lora_err_t lora_idle (lora_struct_t *lora)

  Sets the radio transceiver in idle mode.
- lora_err_t lora_sleep (lora_struct_t *lora)

  Sets the radio transceiver in sleep mode.
- lora_err_t lora_set_receive_mode (lora_struct_t *lora)

  Sets the radio transceiver in receive mode.

- lora_err_t lora_set_tx_power (lora_struct_t *lora, lora_tx_power_t level)

  Configure power level for transmission.
- lora_err_t lora_set_frequency (lora_struct_t *lora, int32_t frequency)

  Set carrier frequency.
- int32_t lora_get_frequency (lora_struct_t *lora)

  Get the frequency set on LoRa.
- lora_err_t lora_set_spreading_factor (lora_struct_t *lora, lora_spreading_factor_t sf)

  Set spreading factor.
- lora_err_t lora_set_bandwidth (lora_struct_t *lora, lora_bandwith_t sbw)

  Set bandwidth (bit rate)
- lora_err_t lora_set_coding_rate (lora_struct_t *lora, int16_t denominator)

  Set coding rate.
- lora_err_t lora_set_preamble_length (lora_struct_t *lora, int32_t length)

  Set the size of preamble.
- lora_err_t lora_set_sync_word (lora_struct_t *lora, int16_t sw)

  Change radio sync word.
- lora_err_t **lora_enable_crc** (lora_struct_t *lora)

  Enable appending/verifying packet CRC.
- lora_err_t **lora_disable_crc** (lora_struct_t *lora)

  Disable appending/verifying packet CRC.
- lora_err_t lora_fill_fifo_buf_to_send (lora_struct_t *lora, uint8_t *buf, int16_t size)

  Fills the REG_FIFO buffer with desired values, and the REG_PAYLOAD_LENGTH with buffer size.
- lora_err_t lora_start_transmission (lora_struct_t *lora)

  Writes the REG_OP_MODE to mode TX.
- bool lora_check_tx_done (lora_struct_t *lora)

  Checks whether the transmission has finished.
- lora_err_t lora_write_irq_flags (lora_struct_t *lora)

  Writes the REG_IRQ_FLAGS buffer with IRQ_TX_DONE_MASK.
- lora_err_t lora_send_packet (lora_struct_t *lora, uint8_t *buf, int16_t size)

  Send a packet. DOES NOT go into receive mode automatically afterwards.
- int16_t lora_receive_packet (lora_struct_t *lora, uint8_t *buf, int16_t size)

  Read a received packet.
- lora_err_t lora_received (lora_struct_t *lora)
- int16_t lora_packet_rssi (lora_struct_t *lora)
- float lora_packet_snr (lora_struct_t *lora)
- void **lora_close** (lora_struct_t *lora)

  Shutdown hardware.
- void **lora_dump_registers** (lora_struct_t *lora)

  Dump registers :D.

4.13.1 Detailed Description

RFM95w LoRa library - multi-MCU.

4.13.2 Function Documentation

4.13.2.1 lora_check_tx_done()

```
bool lora_check_tx_done (
            lora_struct_t * lora )
```

Checks whether the transmission has finished.

Returns

True if finished, false otherwise

4.13.2.2 lora_fill_fifo_buf_to_send()

```
lora_err_t lora_fill_fifo_buf_to_send (
            lora_struct_t * lora,
            uint8_t * buf,
            int16_t size )
```

Fills the REG_FIFO buffer with desired values, and the REG_PAYLOAD_LENGTH with buffer size.

Parameters

buf	- array of 8bit values
size	- sizeof(buf)

Returns

LORA_OK if writing to buffers is ok, LORA_WRITE_ERR otherwise

4.13.2.3 lora_get_frequency()

```
int32_t lora_get_frequency (
            lora_struct_t * lora )
```

Get the frequency set on LoRa.

Returns

LoRa frequency in Hz

4.13.2.4 lora_idle()

```
lora_err_t lora_idle (
            lora_struct_t * lora )
```

Sets the radio transceiver in idle mode.

Note

Must be used to change registers and access the FIFO.

4.13.2.5 lora_implicit_header_mode()

```
lora_err_t lora_implicit_header_mode (
            lora_struct_t * lora,
            int16_t size )
```

Configure implicit header mode. All packets will have a predefined size.

Parameters

size	Size of the packets.

4.13.2.6 lora_packet_rssi()

```
int16_t lora_packet_rssi (
            lora_struct_t * lora )
```

Returns

last packet's RSSI.

4.13.2.7 lora_packet_snr()

```
float lora_packet_snr (
            lora_struct_t * lora )
```

Returns

last packet's SNR (signal to noise ratio).

4.13.2.8 lora_read_reg()

```
uint8_t lora_read_reg (
            lora_struct_t * lora,
            int16_t reg )
```

Read the current value of a register.

Parameters

reg	Register index.

Returns

Value of the register.

4.13.2.9 lora_receive_packet()

```
int16_t lora_receive_packet (
            lora_struct_t * lora,
            uint8_t * buf,
            int16_t size )
```

Read a received packet.

Parameters

buf	Buffer for the data.
size	Available size in buffer (bytes).

Returns

Number of bytes received (zero if no packet available).

4.13.2.10 lora_received()

```
lora_err_t lora_received (
            lora_struct_t * lora )
```

Returns

non-zero if there is data to read (packet received).

4.13.2.11 lora_reset()

```
void lora_reset (
            lora_struct_t * lora )
```

Perform physical reset on the Lora chip.

Exceptions

Assert	if _gpio_set_level fails

4.13.2.12 lora_send_packet()

```
lora_err_t lora_send_packet (
            lora_struct_t * lora,
            uint8_t * buf,
            int16_t size )
```

Send a packet. DOES NOT go into receive mode automatically afterwards.

Parameters

buf	Data to be sent
size	Size of data.

4.13.2.13 lora_set_bandwidth()

```
lora_err_t lora_set_bandwidth (
            lora_struct_t * lora,
            lora_bandwith_t sbw )
```

Set bandwidth (bit rate)

Parameters

sbw	Bandwidth in Hz (up to 500000)

Note

When using low frequency (below 169 MHz), only sf up to 125 kHz is supported

4.13.2.14 lora_set_coding_rate()

```
lora_err_t lora_set_coding_rate (
            lora_struct_t * lora,
            int16_t denominator )
```

Set coding rate.

Parameters

denominator	5-8, Denominator for the coding rate 4/x

4.13.2.15 lora_set_frequency()

```
lora_err_t lora_set_frequency (
            lora_struct_t * lora,
            int32_t frequency )
```

Set carrier frequency.

Parameters

frequency	Frequency in Hz

4.13.2.16 lora_set_preamble_length()

```
lora_err_t lora_set_preamble_length (
            lora_struct_t * lora,
            int32_t length )
```

Set the size of preamble.

Parameters

length	Preamble length in symbols.

4.13.2.17 lora_set_receive_mode()

```
lora_err_t lora_set_receive_mode (
            lora_struct_t * lora )
```

Sets the radio transceiver in receive mode.

Note

Incoming packets will be received.

4.13.2.18 lora_set_spreading_factor()

```
lora_err_t lora_set_spreading_factor (
            lora_struct_t * lora,
            lora_spreading_factor_t sf )
```

Set spreading factor.

Parameters

sf	6-12, Spreading factor to use.

Returns

LORA_OK if operation successful, LORA_CONFIG_ERR otherwise

4.13.2.19 lora_set_sync_word()

```
lora_err_t lora_set_sync_word (
            lora_struct_t * lora,
            int16_t sw )
```

Change radio sync word.

Parameters

sw	New sync word to use.

4.13.2.20 lora_set_tx_power()

```
lora_err_t lora_set_tx_power (
            lora_struct_t * lora,
            lora_tx_power_t level )
```

Configure power level for transmission.

Parameters

level	2 or 17, from least to most power

4.13.2.21 lora_sleep()

```
lora_err_t lora_sleep (
            lora_struct_t * lora )
```

Sets the radio transceiver in sleep mode.

Note

Low power consumption and FIFO is lost.

4.13.2.22 lora_start_transmission()

```
lora_err_t lora_start_transmission (
            lora_struct_t * lora )
```

Writes the REG_OP_MODE to mode TX.

Returns

LORA_OK if all goes good, LORA_WRITE_ERR otherwise

4.13.2.23 lora_write_irq_flags()

```
lora_err_t lora_write_irq_flags (
            lora_struct_t * lora )
```

Writes the REG_IRQ_FLAGS buffer with IRQ_TX_DONE_MASK.

Returns

LORA_OK :D - LORA_WRITE_ERR :C

4.13.2.24 lora_write_reg()

```
lora_err_t lora_write_reg (
            lora_struct_t * lora,
            int16_t reg,
            int16_t val )
```

Write a value to a register.

Parameters

reg	Register index.
val	Value to write.

Returns

lora_err_t value

4.14 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/lora/lora.h File Reference

RFM95w LoRa library - multi-MCU.

```
#include <stdint.h>
#include <string.h>
#include <stdio.h>
#include <stdbool.h>
```

Classes

- struct lora_struct_t

Macros

- #define **IRQ_TX_DONE_MASK** 0x08
- #define **IRQ_PAYLOAD_CRC_ERROR_MASK** 0x20
- #define **IRQ_RX_DONE_MASK** 0x40
- #define **PA_OUTPUT_RFO_PIN** 0
- #define **PA_OUTPUT_PA_BOOST_PIN** 1
- #define **TIMEOUT_RESET** 100
- #define **REG_FIFO** 0x00
- #define **REG_OP_MODE** 0x01
- #define **REG_FRF_MSB** 0x06
- #define **REG_FRF_MID** 0x07
- #define **REG_FRF_LSB** 0x08
- #define **REG_PA_CONFIG** 0x09
- #define **REG_LNA** 0x0c
- #define **REG_FIFO_ADDR_PTR** 0x0d
- #define **REG_FIFO_TX_BASE_ADDR** 0x0e

- #define **REG_FIFO_RX_BASE_ADDR** 0x0f
- #define **REG_FIFO_RX_CURRENT_ADDR** 0x10
- #define **REG_IRQ_FLAGS** 0x12
- #define **REG_RX_NB_BYTES** 0x13
- #define **REG_PKT_SNR_VALUE** 0x19
- #define **REG_PKT_RSSI_VALUE** 0x1a
- #define **REG_MODEM_CONFIG_1** 0x1d
- #define **REG_MODEM_CONFIG_2** 0x1e
- #define **REG_PREAMBLE_MSB** 0x20
- #define **REG_PREAMBLE_LSB** 0x21
- #define **REG_PAYLOAD_LENGTH** 0x22
- #define **REG_MODEM_CONFIG_3** 0x26
- #define **REG_RSSI_WIDEBAND** 0x2c
- #define **REG_DETECTION_OPTIMIZE** 0x31
- #define **REG_DETECTION_THRESHOLD** 0x37
- #define **REG_SYNC_WORD** 0x39
- #define **REG_DIO_MAPPING_1** 0x40
- #define **REG_VERSION** 0x42
- #define **MODE_LONG_RANGE_MODE** 0x80
- #define **MODE_SLEEP** 0x00
- #define **MODE_STDBY** 0x01
- #define **MODE_TX** 0x03
- #define **MODE_RX_CONTINUOUS** 0x05
- #define **MODE_RX_SINGLE** 0x06
- #define **PA_BOOST** 0x80

Typedefs

- typedef bool(∗ **lora_SPI_transmit**) (uint8_t _in[2], uint8_t _val[2])
- typedef void(∗ **lora_delay**) (size_t _ms)
- typedef bool(∗ **lora_GPIO_set_level**) (uint8_t _gpio_num, uint32_t _level)
- typedef void(∗ **lora_log**) (const char ∗info)

Enumerations

- enum lora_err_t {
  LORA_OK = 0 , **LORA_INIT_ERR** , **LORA_WRITE_ERR** , **LORA_TRANSMIT_ERR** ,
  LORA_RECEIVE_ERR , **LORA_CONFIG_ERR** }

  Lora functions return values enum.
- enum **lora_gpio_mode_t** { **LORA_GPIO_MODE_DISABLE** = 0 , **LORA_GPIO_MODE_INPUT** , **LORA_↩ GPIO_MODE_OUTPUT** }
- enum lora_bandwith_t {
  LORA_BW_7_8_kHz = 0 , **LORA_BW_10_4_kHz** , **LORA_BW_15_6_kHz** , **LORA_BW_20_8_kHz** ,
  LORA_BW_31_25_kHz , **LORA_BW_41_7_kHz** , **LORA_BW_62_5_kHz** , **LORA_BW_125_kHz** ,
  LORA_BW_250_kHz , **LORA_BW_500_kHz** }

  Enum for LoRa bandwith in Hz.
- enum lora_spreading_factor_t {
  LORA_SF_64_CoS = 6 , **LORA_SF_128_CoS** , **LORA_SF_256_CoS** , **LORA_SF_512_CoS** ,
  LORA_SF_1024_CoS , **LORA_SF_2048_CoS** , **LORA_SF_4096_CoS** }

  Enum for LoRa spreading factor in chips / symbol Used in the lora_set_spreading_factor method.
- enum lora_tx_power_t { **LORA_TX_POWER_14_dBm** = 2 , **LORA_TX_POWER_20_dBm** = 17 }

  Enum for LoRa TX Power.

Functions

- lora_err_t **lora_init** (lora_struct_t *lora)

  Perform hardware initialization.
- lora_err_t **lora_default_config** (lora_struct_t *lora)

  Create default config and set parameters.
- lora_err_t lora_write_reg (lora_struct_t *lora, int16_t reg, int16_t val)

  Write a value to a register.
- uint8_t lora_read_reg (lora_struct_t *lora, int16_t reg)

  Read the current value of a register.
- void lora_reset (lora_struct_t *lora)

  Perform physical reset on the Lora chip.
- lora_err_t **lora_explicit_header_mode** (lora_struct_t *lora)

  Configure explicit header mode. Packet size will be included in the frame.
- lora_err_t lora_implicit_header_mode (lora_struct_t *lora, int16_t size)

  Configure implicit header mode. All packets will have a predefined size.
- lora_err_t lora_idle (lora_struct_t *lora)

  Sets the radio transceiver in idle mode.
- lora_err_t lora_sleep (lora_struct_t *lora)

  Sets the radio transceiver in sleep mode.
- lora_err_t lora_set_receive_mode (lora_struct_t *lora)

  Sets the radio transceiver in receive mode.
- lora_err_t lora_set_tx_power (lora_struct_t *lora, lora_tx_power_t level)

  Configure power level for transmission.
- lora_err_t lora_set_frequency (lora_struct_t *lora, int32_t frequency)

  Set carrier frequency.
- int32_t lora_get_frequency (lora_struct_t *lora)

  Get the frequency set on LoRa.
- lora_err_t lora_set_spreading_factor (lora_struct_t *lora, lora_spreading_factor_t sf)

  Set spreading factor.
- lora_err_t lora_set_bandwidth (lora_struct_t *lora, lora_bandwith_t sbw)

  Set bandwidth (bit rate)
- lora_err_t lora_set_coding_rate (lora_struct_t *lora, int16_t denominator)

  Set coding rate.
- lora_err_t lora_set_preamble_length (lora_struct_t *lora, int32_t length)

  Set the size of preamble.
- lora_err_t lora_set_sync_word (lora_struct_t *lora, int16_t sw)

  Change radio sync word.
- lora_err_t **lora_enable_crc** (lora_struct_t *lora)

  Enable appending/verifying packet CRC.
- lora_err_t **lora_disable_crc** (lora_struct_t *lora)

  Disable appending/verifying packet CRC.
- lora_err_t lora_fill_fifo_buf_to_send (lora_struct_t *lora, uint8_t *buf, int16_t size)

  Fills the REG_FIFO buffer with desired values, and the REG_PAYLOAD_LENGTH with buffer size.
- lora_err_t lora_start_transmission (lora_struct_t *lora)

  Writes the REG_OP_MODE to mode TX.
- bool lora_check_tx_done (lora_struct_t *lora)

  Checks whether the transmission has finished.
- lora_err_t lora_write_irq_flags (lora_struct_t *lora)

  Writes the REG_IRQ_FLAGS buffer with IRQ_TX_DONE_MASK.
- lora_err_t lora_send_packet (lora_struct_t *lora, uint8_t *buf, int16_t size)

Send a packet. DOES NOT go into receive mode automatically afterwards.

- int16_t lora_receive_packet (lora_struct_t *lora, uint8_t *buf, int16_t size)

  Read a received packet.
- lora_err_t lora_received (lora_struct_t *lora)
- int16_t lora_packet_rssi (lora_struct_t *lora)
- float lora_packet_snr (lora_struct_t *lora)
- void **lora_close** (lora_struct_t *lora)

  Shutdown hardware.
- int16_t **lora_initialized** (lora_struct_t *lora)

  Not supported.
- void **lora_dump_registers** (lora_struct_t *lora)

  Dump registers :D.

4.14.1 Detailed Description

RFM95w LoRa library - multi-MCU.

4.14.2 Function Documentation

4.14.2.1 lora_check_tx_done()

```
bool lora_check_tx_done (
            lora_struct_t * lora )
```

Checks whether the transmission has finished.

Returns

True if finished, false otherwise

4.14.2.2 lora_fill_fifo_buf_to_send()

```
lora_err_t lora_fill_fifo_buf_to_send (
            lora_struct_t * lora,
            uint8_t * buf,
            int16_t size )
```

Fills the REG_FIFO buffer with desired values, and the REG_PAYLOAD_LENGTH with buffer size.

Parameters

buf	- array of 8bit values
size	- sizeof(buf)

Returns

LORA_OK if writing to buffers is ok, LORA_WRITE_ERR otherwise

4.14.2.3 lora_get_frequency()

```
int32_t lora_get_frequency (
            lora_struct_t * lora )
```

Get the frequency set on LoRa.

Returns

LoRa frequency in Hz

4.14.2.4 lora_idle()

```
lora_err_t lora_idle (
            lora_struct_t * lora )
```

Sets the radio transceiver in idle mode.

Note

Must be used to change registers and access the FIFO.

4.14.2.5 lora_implicit_header_mode()

```
lora_err_t lora_implicit_header_mode (
            lora_struct_t * lora,
            int16_t size )
```

Configure implicit header mode. All packets will have a predefined size.

Parameters

size	Size of the packets.

4.14.2.6 lora_packet_rssi()

```
int16_t lora_packet_rssi (
            lora_struct_t * lora )
```

Returns

last packet's RSSI.

4.14.2.7 lora_packet_snr()

```
float lora_packet_snr (
            lora_struct_t * lora )
```

Returns

last packet's SNR (signal to noise ratio).

4.14.2.8 lora_read_reg()

```
uint8_t lora_read_reg (
            lora_struct_t * lora,
            int16_t reg )
```

Read the current value of a register.

Parameters

reg	Register index.

Returns

Value of the register.

4.14.2.9 lora_receive_packet()

```
int16_t lora_receive_packet (
            lora_struct_t * lora,
            uint8_t * buf,
            int16_t size )
```

Read a received packet.

Parameters

buf	Buffer for the data.
size	Available size in buffer (bytes).

Returns

Number of bytes received (zero if no packet available).

4.14.2.10 lora_received()

```
lora_err_t lora_received (
            lora_struct_t * lora )
```

Returns

non-zero if there is data to read (packet received).

4.14.2.11 lora_reset()

```
void lora_reset (
            lora_struct_t * lora )
```

Perform physical reset on the Lora chip.

Exceptions

Assert	if _gpio_set_level fails

4.14.2.12 lora_send_packet()

```
lora_err_t lora_send_packet (
            lora_struct_t * lora,
            uint8_t * buf,
            int16_t size )
```

Send a packet. DOES NOT go into receive mode automatically afterwards.

Parameters

buf	Data to be sent
size	Size of data.

4.14.2.13 lora_set_bandwidth()

```
lora_err_t lora_set_bandwidth (
            lora_struct_t * lora,
            lora_bandwith_t sbw )
```

Set bandwidth (bit rate)

Parameters

sbw	Bandwidth in Hz (up to 500000)

Note

When using low frequency (below 169 MHz), only sf up to 125 kHz is supported

4.14.2.14 lora_set_coding_rate()

```
lora_err_t lora_set_coding_rate (
            lora_struct_t * lora,
            int16_t denominator )
```

Set coding rate.

Parameters

denominator	5-8, Denominator for the coding rate 4/x

4.14.2.15 lora_set_frequency()

```
lora_err_t lora_set_frequency (
            lora_struct_t * lora,
            int32_t frequency )
```

Set carrier frequency.

Parameters

frequency	Frequency in Hz

4.14.2.16 lora_set_preamble_length()

```
lora_err_t lora_set_preamble_length (
            lora_struct_t * lora,
            int32_t length )
```

Set the size of preamble.

Parameters

length	Preamble length in symbols.

4.14.2.17 lora_set_receive_mode()

```
lora_err_t lora_set_receive_mode (
            lora_struct_t * lora )
```

Sets the radio transceiver in receive mode.

Note

Incoming packets will be received.

4.14.2.18 lora_set_spreading_factor()

```
lora_err_t lora_set_spreading_factor (
            lora_struct_t * lora,
            lora_spreading_factor_t sf )
```

Set spreading factor.

Parameters

sf	6-12, Spreading factor to use.

Returns

LORA_OK if operation successful, LORA_CONFIG_ERR otherwise

4.14.2.19 lora_set_sync_word()

```
lora_err_t lora_set_sync_word (
            lora_struct_t * lora,
            int16_t sw )
```

Change radio sync word.

Parameters

sw	New sync word to use.

4.14.2.20 lora_set_tx_power()

```
lora_err_t lora_set_tx_power (
            lora_struct_t * lora,
            lora_tx_power_t level )
```

Configure power level for transmission.

Parameters

level	2 or 17, from least to most power

4.14.2.21 lora_sleep()

```
lora_err_t lora_sleep (
            lora_struct_t * lora )
```

Sets the radio transceiver in sleep mode.

Note

Low power consumption and FIFO is lost.

4.14.2.22 lora_start_transmission()

```
lora_err_t lora_start_transmission (
            lora_struct_t * lora )
```

Writes the REG_OP_MODE to mode TX.

Returns

LORA_OK if all goes good, LORA_WRITE_ERR otherwise

4.14.2.23 lora_write_irq_flags()

```
lora_err_t lora_write_irq_flags (
            lora_struct_t * lora )
```

Writes the REG_IRQ_FLAGS buffer with IRQ_TX_DONE_MASK.

Returns

LORA_OK :D - LORA_WRITE_ERR :C

4.14.2.24 lora_write_reg()

```
lora_err_t lora_write_reg (
            lora_struct_t * lora,
            int16_t reg,
            int16_t val )
```

Write a value to a register.

Parameters

reg	Register index.
val	Value to write.

Returns

lora_err_t value

4.15 lora.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002 #pragma once
00003
00004 #include <stdint.h>
00005 #include <string.h>
00006 #include <stdio.h>
00007 #include <stdbool.h>
```

```
00008
00014 /*
00015  * IRQ masks
00016  */
00017 #define IRQ_TX_DONE_MASK 0x08
00018 #define IRQ_PAYLOAD_CRC_ERROR_MASK 0x20
00019 #define IRQ_RX_DONE_MASK 0x40
00020
00021 #define PA_OUTPUT_RFO_PIN 0
00022 #define PA_OUTPUT_PA_BOOST_PIN 1
00023
00024 #define TIMEOUT_RESET 100
00025
00026 /*
00027  * Register definitions
00028  */
00029 #define REG_FIFO 0x00
00030 #define REG_OP_MODE 0x01
00031 #define REG_FRF_MSB 0x06
00032 #define REG_FRF_MID 0x07
00033 #define REG_FRF_LSB 0x08
00034 #define REG_PA_CONFIG 0x09
00035 #define REG_LNA 0x0c
00036 #define REG_FIFO_ADDR_PTR 0x0d
00037 #define REG_FIFO_TX_BASE_ADDR 0x0e
00038 #define REG_FIFO_RX_BASE_ADDR 0x0f
00039 #define REG_FIFO_RX_CURRENT_ADDR 0x10
00040 #define REG_IRQ_FLAGS 0x12
00041 #define REG_RX_NB_BYTES 0x13
00042 #define REG_PKT_SNR_VALUE 0x19
00043 #define REG_PKT_RSSI_VALUE 0x1a
00044 #define REG_MODEM_CONFIG_1 0x1d
00045 #define REG_MODEM_CONFIG_2 0x1e
00046 #define REG_PREAMBLE_MSB 0x20
00047 #define REG_PREAMBLE_LSB 0x21
00048 #define REG_PAYLOAD_LENGTH 0x22
00049 #define REG_MODEM_CONFIG_3 0x26
00050 #define REG_RSSI_WIDEBAND 0x2c
00051 #define REG_DETECTION_OPTIMIZE 0x31
00052 #define REG_DETECTION_THRESHOLD 0x37
00053 #define REG_SYNC_WORD 0x39
00054 #define REG_DIO_MAPPING_1 0x40
00055 #define REG_VERSION 0x42
00056
00057 /*
00058  * Transceiver modes
00059  */
00060 #define MODE_LONG_RANGE_MODE 0x80
00061 #define MODE_SLEEP 0x00
00062 #define MODE_STDBY 0x01
00063 #define MODE_TX 0x03
00064 #define MODE_RX_CONTINUOUS 0x05
00065 #define MODE_RX_SINGLE 0x06
00066
00067 /*
00068  * PA configuration
00069  */
00070 #define PA_BOOST 0x80
00071
00075 typedef enum {
00076   LORA_OK = 0,
00077   LORA_INIT_ERR,
00078   LORA_WRITE_ERR,
00079   LORA_TRANSMIT_ERR,
00080   LORA_RECEIVE_ERR,
00081   LORA_CONFIG_ERR
00082 } lora_err_t;
00083
00084 typedef enum {
00085   LORA_GPIO_MODE_DISABLE = 0,
00086   LORA_GPIO_MODE_INPUT,
00087   LORA_GPIO_MODE_OUTPUT
00088 } lora_gpio_mode_t;
00089
00093 typedef enum {
00094   LORA_BW_7_8_kHz = 0,
00095   LORA_BW_10_4_kHz,
00096   LORA_BW_15_6_kHz,
00097   LORA_BW_20_8_kHz,
00098   LORA_BW_31_25_kHz,
00099   LORA_BW_41_7_kHz,
00100   LORA_BW_62_5_kHz,
00101   LORA_BW_125_kHz,
00102   LORA_BW_250_kHz,
00103   LORA_BW_500_kHz,
00104 } lora_bandwith_t;
00105
```

```
00110 typedef enum {
00111   LORA_SF_64_CoS = 6,
00112   LORA_SF_128_CoS,
00113   LORA_SF_256_CoS,
00114   LORA_SF_512_CoS,
00115   LORA_SF_1024_CoS,
00116   LORA_SF_2048_CoS,
00117   LORA_SF_4096_CoS
00118 } lora_spreading_factor_t;
00119
00123 // TODO(Glibus): check if it is ok
00124 typedef enum {
00125   LORA_TX_POWER_14_dBm = 2,
00126   LORA_TX_POWER_20_dBm = 17
00127 } lora_tx_power_t;
00128
00129 typedef bool (*lora_SPI_transmit)(uint8_t _in[2], uint8_t _val[2]);
00130 typedef void (*lora_delay)(size_t _ms);
00131 typedef bool (*lora_GPIO_set_level)(uint8_t _gpio_num, uint32_t _level);
00132 typedef void (*lora_log)(const char *info);
00133
00134 typedef struct {
00135   lora_SPI_transmit _spi_transmit;
00136   lora_delay _delay;
00137   lora_GPIO_set_level _gpio_set_level;
00138   lora_log log;
00139   uint8_t rst_gpio_num;
00140   uint8_t cs_gpio_num;
00141   uint8_t d0_gpio_num;
00142   int16_t implicit_header;
00143   int32_t frequency;
00144 } lora_struct_t;
00145
00149 lora_err_t lora_init(lora_struct_t *lora);
00150
00154 lora_err_t lora_default_config(lora_struct_t *lora);
00155
00162 lora_err_t lora_write_reg(lora_struct_t *lora, int16_t reg, int16_t val);
00163
00169 uint8_t lora_read_reg(lora_struct_t *lora, int16_t reg);
00170
00175 void lora_reset(lora_struct_t *lora);
00176
00181 lora_err_t lora_explicit_header_mode(lora_struct_t *lora);
00182
00188 lora_err_t lora_implicit_header_mode(lora_struct_t *lora, int16_t size);
00189
00194 lora_err_t lora_idle(lora_struct_t *lora);
00195
00200 lora_err_t lora_sleep(lora_struct_t *lora);
00201
00206 lora_err_t lora_set_receive_mode(lora_struct_t *lora);
00207
00212 lora_err_t lora_set_tx_power(lora_struct_t *lora, lora_tx_power_t level);
00213
00218 lora_err_t lora_set_frequency(lora_struct_t *lora, int32_t frequency);
00219
00224 int32_t lora_get_frequency(lora_struct_t *lora);
00225
00231 lora_err_t lora_set_spreading_factor(lora_struct_t *lora,
00232                                      lora_spreading_factor_t sf);
00233
00240 lora_err_t lora_set_bandwidth(lora_struct_t *lora, lora_bandwith_t sbw);
00241
00246 lora_err_t lora_set_coding_rate(lora_struct_t *lora, int16_t denominator);
00247
00252 lora_err_t lora_set_preamble_length(lora_struct_t *lora, int32_t length);
00253
00258 lora_err_t lora_set_sync_word(lora_struct_t *lora, int16_t sw);
00259
00263 lora_err_t lora_enable_crc(lora_struct_t *lora);
00264
00268 lora_err_t lora_disable_crc(lora_struct_t *lora);
00269
00278 lora_err_t lora_fill_fifo_buf_to_send(lora_struct_t *lora, uint8_t *buf,
00279                                       int16_t size);
00280
00285 lora_err_t lora_start_transmission(lora_struct_t *lora);
00286
00291 bool lora_check_tx_done(lora_struct_t *lora);
00292
00297 lora_err_t lora_write_irq_flags(lora_struct_t *lora);
00298
00304 lora_err_t lora_send_packet(lora_struct_t *lora, uint8_t *buf, int16_t size);
00305
00312 int16_t lora_receive_packet(lora_struct_t *lora, uint8_t *buf, int16_t size);
00313
```

```
00317 lora_err_t lora_received(lora_struct_t *lora);
00318
00322 int16_t lora_packet_rssi(lora_struct_t *lora);
00323
00327 float lora_packet_snr(lora_struct_t *lora);
00328
00332 void lora_close(lora_struct_t *lora);
00333
00335 int16_t lora_initialized(lora_struct_t *lora);
00336
00338 void lora_dump_registers(lora_struct_t *lora);
```

4.16 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/mcu_config/lora_esp32_config.c File Reference

ESP32 lora configuration.

```
#include "lora_esp32_config.h"
```

Macros

- #define **TAG** "LORA"

Functions

- bool **_lora_spi_and_pins_init** ()
- bool **_lora_SPI_transmit** (uint8_t _in[2], uint8_t _out[2])
- void **_lora_delay** (size_t _ms)
- bool **_lora_GPIO_set_level** (uint8_t _gpio_num, uint32_t _level)
- void **_lora_log** (const char *info)

4.16.1 Detailed Description

ESP32 lora configuration.

4.17 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/mcu_config/lora_esp32_config.h File Reference

ESP32 lora configuration.

```
#include <stdbool.h>
#include <stdint.h>
#include "driver/gpio.h"
#include "driver/spi_master.h"
#include "esp_log.h"
#include "esp_rom_gpio.h"
#include "freertos/task.h"
#include "sdkconfig.h"
#include "rom/gpio.h"
#include "soc/gpio_struct.h"
```

Functions

- bool **_lora_spi_and_pins_init** ()
- bool **_lora_SPI_transmit** (uint8_t _in[2], uint8_t _out[2])
- void **_lora_delay** (size_t _ms)
- bool **_lora_GPIO_set_level** (uint8_t _gpio_num, uint32_t _level)
- void **_lora_log** (const char *info)

4.17.1 Detailed Description

ESP32 lora configuration.

4.18 lora_esp32_config.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002 #pragma once
00003
00004 #include <stdbool.h>
00005 #include <stdint.h>
00006
00007 #include "driver/gpio.h"
00008 #include "driver/spi_master.h"
00009 #include "esp_log.h"
00010 #include "esp_rom_gpio.h"
00011 #include "freertos/task.h"
00012 #include "sdkconfig.h"
00013 #include "rom/gpio.h"
00014 #include "soc/gpio_struct.h"
00015
00021 bool _lora_spi_and_pins_init();
00022
00023 bool _lora_SPI_transmit(uint8_t _in[2], uint8_t _out[2]);
00024
00025 void _lora_delay(size_t _ms);
00026
00027 bool _lora_GPIO_set_level(uint8_t _gpio_num, uint32_t _level);
00028
00029 void _lora_log(const char* info);
```

4.19 mcu_adc_config.h

```
00001 // Copyright 2023 PWrInSpace, Krzysztof Gliwiński
00002
00003 #pragma once
00004 #include <stdbool.h>
00005 #include <stdint.h>
00006
00007 #include "esp_adc/adc_cali.h"
00008 #include "esp_adc/adc_cali_scheme.h"
00009 #include "esp_adc/adc_oneshot.h"
00010 #include "esp_log.h"
00011 #include "soc/adc_channel.h"
00017 #define READ_ERROR_RETURN_VAL 0xFFFF
00018 #define VOLTAGE_READ_ERROR_RETURN_VAL -1.0f
00019 #define MAX_ADC_CHANNELS 8
00020
00037 typedef struct {
00038   float adc_cal[MAX_ADC_CHANNELS];
00039   uint8_t adc_chan[MAX_ADC_CHANNELS];
00040   uint8_t adc_chan_num;
00041   adc_oneshot_unit_init_cfg_t oneshot_unit_cfg;
00042   adc_oneshot_chan_cfg_t oneshot_chan_cfg;
00043   adc_oneshot_unit_handle_t* oneshot_unit_handle;
00044 } voltage_measure_config_t;
00045
00053 esp_err_t voltage_measure_init(voltage_measure_config_t* v_mes);
00054
00060 int voltage_measure_read_raw(voltage_measure_config_t* v_mes, uint8_t adc_chan);
00061
00067 float voltage_measure_read_voltage(voltage_measure_config_t* v_mes,
00068                                    uint8_t adc_chan);
```

4.20 mcu_i2c_config.h

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #include "driver/i2c.h"
00004 #include "esp_log.h"
00005 #include "freertos/FreeRTOS.h"
00006 #include "freertos/task.h"
00007 #include "sdkconfig.h"
00008
00009 typedef struct {
00010   i2c_port_t port;
00011   i2c_cmd_handle_t *cmd;
00012   gpio_num_t sda;
00013   gpio_num_t scl;
00014   uint32_t clk_speed;
00015   uint32_t timeout;
00016   bool i2c_init_flag;
00017 } mcu_i2c_config_t;
00018
00025 esp_err_t i2c_init(mcu_i2c_config_t *i2c);
```

4.21 mcu_spi_config.h

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #pragma once
00004
00005 #include <stdbool.h>
00006
00007 #include "driver/gpio.h"
00008 #include "driver/spi_master.h"
00009 #include "esp_log.h"
00010 #include "esp_rom_gpio.h"
00011 #include "freertos/task.h"
00012 #include "rom/gpio.h"
00013 #include "sdkconfig.h"
00014 #include "soc/gpio_struct.h"
00015
00016 typedef struct {
00017   spi_device_handle_t spi;
00018   spi_bus_config_t bus_config;
00019   spi_device_interface_config_t dev_config;
00020   bool spi_init_flag;
00021 } mcu_spi_config_t;
00022
00030 esp_err_t spi_init(mcu_spi_config_t *spi);
```

4.22 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/mcu_config/mcu_twai_config.h File Reference

TWAI configuration and tools.

```
#include <stdint.h>
#include "driver/gpio.h"
#include "driver/twai.h"
#include "esp_err.h"
#include "esp_log.h"
#include "freertos/FreeRTOS.h"
#include "freertos/queue.h"
```

Classes

- struct twai_config_t

  TWAI configuration structure.

Functions

- esp_err_t twai_init (twai_config_t *config, twai_general_config_t *g_config, twai_timing_config_t *t_config, twai_filter_config_t *f_config)

  TWAI initialization, initializes and starts TWAI driver.
- twai_message_t compose_self_test_message (uint32_t id, uint8_t data_length_code, uint8_t *data)

  TWAI message composition for self test purposes.

4.22.1 Detailed Description

TWAI configuration and tools.

4.22.2 Function Documentation

4.22.2.1 compose_self_test_message()

```
twai_message_t compose_self_test_message (
            uint32_t id,
            uint8_t data_length_code,
            uint8_t * data )
```

TWAI message composition for self test purposes.

Parameters

id	Message ID
data_length_code	Message data length code
data	Pointer to message data

4.22.2.2 twai_init()

```
esp_err_t twai_init (
            twai_config_t * config,
            twai_general_config_t * g_config,
            twai_timing_config_t * t_config,
            twai_filter_config_t * f_config )
```

TWAI initialization, initializes and starts TWAI driver.

Note

Parameters

config	TWAI configuration structure

Returns

ESP_OK on success, ESP_FAIL otherwise

4.23 mcu_twai_config.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #pragma once
00004
00010 #include <stdint.h>
00011
00012 #include "driver/gpio.h"
00013 #include "driver/twai.h"
00014 #include "esp_err.h"
00015 #include "esp_log.h"
00016 #include "freertos/FreeRTOS.h"
00017 #include "freertos/queue.h"
00018
00026 typedef struct {
00027   gpio_num_t tx_gpio_num;
00028   gpio_num_t rx_gpio_num;
00029   twai_mode_t mode;
00030 } twai_config_t;
00031
00038 esp_err_t twai_init(twai_config_t *config, twai_general_config_t *g_config,
00039                     twai_timing_config_t *t_config,
00040                     twai_filter_config_t *f_config);
00041
00048 twai_message_t compose_self_test_message(uint32_t id, uint8_t data_length_code,
00049                                          uint8_t *data);
```

4.24 ssd1306_esp32_config.h

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #include "mcu_i2c_config.h"
00004 #include "ssd1306.h"
00005
00006 void ssd1306_esp32_config_mount_i2c_config(mcu_i2c_config_t* _i2c_config);
00007
00008 bool _ssd1306_i2c_master_write_byte(ssd1306_i2c_cmd_handle_t cmd, uint8_t _data,
00009                                     bool _ack_en);
00010
00011 bool _ssd1306_i2c_master_write(ssd1306_i2c_cmd_handle_t cmd,
00012                                const uint8_t* _data, size_t _data_len,
00013                                bool _ack_en);
00014
00015 bool _ssd1306_i2c_master_start(ssd1306_i2c_cmd_handle_t cmd);
00016
00017 bool _ssd1306_i2c_master_stop(ssd1306_i2c_cmd_handle_t cmd);
00018
00019 bool _ssd1306_i2c_master_cmd_begin(ssd1306_i2c_cmd_handle_t cmd,
00020                                    uint16_t ticks_to_wait);
00021
00022 // TODO(Glibus): tu sie moze wyjebac
00023 ssd1306_i2c_cmd_handle_t _ssd1306_i2c_cmd_link_create();
00024
00025 void _ssd1306_i2c_cmd_link_delete(ssd1306_i2c_cmd_handle_t cmd);
00026
00027 void _ssd1306_delay(size_t _ms);
00028
00029 void _ssd1306_log(const ssd1306_log_level_t level, const char* tag, char* info);
```

4.25 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/memory/flash.h File Reference

contains API to store huge data files in ESP internal flash. Description based on https://esp32tutorials.↩ com/esp32-spiffs-esp-idf/

```
#include <stddef.h>
#include <stdint.h>
#include <string.h>
#include "esp_flash.h"
#include "esp_flash_spi_init.h"
#include "esp_log.h"
#include "esp_spiffs.h"
#include "spi_flash_mmap.h"
```

Macros

- #define **PATH** "/spiffs"
- #define **FLASH_FILE_NAME** "flash"
- #define **MAX_FILES** 1
- #define **FLASH_PATH** PATH "/" FLASH_FILE_NAME

Enumerations

- enum **FlashResult** {
  FLASH_OK , **FLASH_INIT_ERROR** , **FLASH_ALREADY_INITIALIZED** , **FLASH_IS_NOT_INITIALIZED** ,
  FLASH_CHECK_FAILED , **FLASH_WRITE_ERROR** , **FLASH_OPEN_ERROR** , **FLASH_FORMAT_↩ ERROR** ,
  FLASH_READ_NO_DATA , **FLASH_READ_ALL_SIZE** , **FLASH_ALLOC_ERROR** }

Functions

- FlashResult **FLASH_init** (uint8_t max_files)
- FlashResult **FLASH_write** (const char ∗file_name, const char ∗data, size_t size)
- FlashResult **FLASH_read_all_data** (const char ∗file_name, char ∗data_container, size_t size)
- size_t **FLASH_get_used_size** (void)
- size_t **FLASH_get_total_size** (void)
- FlashResult **FLASH_format** (void)

4.25.1 Detailed Description

contains API to store huge data files in ESP internal flash. Description based on https://esp32tutorials.↩ com/esp32-spiffs-esp-idf/

4.25.2 to create a spiffs partition in flash:

4.25.2.1 a file and name it 'partitions.csv' e.g.:

4.25.3 Name, Type, SubType, Offset, Size, Flags

4.25.4 Note: if you change the phy_init or app partition offset,

make sure to change the offset in Kconfig.projbuild nvs, data, nvs, , 0x6000, phy_init, data, phy, , 0x1000, factory, app, factory, , 1M, storage, data, spiffs, , 1M

4.25.4.1 create a folder named spiffs data and include files e.g.

data.txt.

4.25.4.2 create a folder named spiffs data and include files e.g.

spiffs_create_partition_image(storage ../spiffs_data FLASH_IN_PROJECT)

4.25.4.3 idf.py menuconfig, and go to > Serial flasher condif

4.25.4.4 to partition table and make sure that these are set:

Flash SPI mode (DIO) —> Flash Sampling Mode (STR Mode) —> Flash SPI speed (40 MHz) —> Flash size (YOUR MCUS FLASH SIZE HERE) —>

4.25.4.5 go to back, and to Partition Table, select these

(partitions.csv) Custom partition CSV file (0x8000) Offset of partition table [*] Generate an MD5 checksum for the partition table ! NOTE: if you change the phy_init or app partition offset, make sure to change the partition table offset (above) accordingly

4.26 flash.h

Go to the documentation of this file.

```
00001 // Copyright 2022 Pwr in Space, Krzysztof Gliwiński
00002 #pragma once
00003
00004 #include <stddef.h>
00005 #include <stdint.h>
00006 #include <string.h>
00007
00008 #include "esp_flash.h"
00009 #include "esp_flash_spi_init.h"
00010 #include "esp_log.h"
00011 #include "esp_spiffs.h"
00012 #include "spi_flash_mmap.h"
00013
00014 #define PATH "/spiffs"
00015 #define FLASH_FILE_NAME "flash"
00016 #define MAX_FILES 1
00017
00018 #define FLASH_PATH PATH "/" FLASH_FILE_NAME
00019
00054 typedef enum {
00055   FLASH_OK,
00056   FLASH_INIT_ERROR,
00057   FLASH_ALREADY_INITIALIZED,
00058   FLASH_IS_NOT_INITIALIZED,
00059   FLASH_CHECK_FAILED,
00060   FLASH_WRITE_ERROR,
00061   FLASH_OPEN_ERROR,
00062   FLASH_FORMAT_ERROR,
00063   FLASH_READ_NO_DATA,
00064   FLASH_READ_ALL_SIZE,
00065   FLASH_ALLOC_ERROR,
00066 } FlashResult;
00067
00068 FlashResult FLASH_init(uint8_t max_files);
00069 FlashResult FLASH_write(const char* file_name, const char* data, size_t size);
00070 FlashResult FLASH_read_all_data(const char* file_name, char* data_container,
00071                                 size_t size);
00072 size_t FLASH_get_used_size(void);
00073 size_t FLASH_get_total_size(void);
00074 FlashResult FLASH_format(void);
```

4.27 flash_nvs.h

```
00001 // Copyright 2022 Pwr in Space, Krzysztof Gliwiński
00002 #pragma once
00003
00004 #include <stdint.h>
00005
00006 #include "esp_log.h"
00007 #include "nvs_flash.h"
00008
00009 typedef enum {
00010   NVS_OK,
00011   NVS_INIT_ERROR,
00012   NVS_OPEN_ERROR,
00013   NVS_READ_ERROR,
00014 } NVSResult;
00015
00021 NVSResult NVS_init(void);
00022
00031 NVSResult NVS_write_uint8(const char* key, uint8_t val);
00032
00041 NVSResult NVS_read_uint8(const char* key, uint8_t* val);
00042
00050 NVSResult NVS_write_uint16(const char* key, uint16_t val);
00051
00059 NVSResult NVS_read_uint16(const char* key, uint16_t* val);
```

4.28 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/misc/led_driver.h File Reference

LED driver for ESP32.

```
#include <stdint.h>
#include "driver/ledc.h"
#include "esp_err.h"
#include "esp_log.h"
#include "stdbool.h"
```

Classes

- struct led_driver_t

  Intended for LEDs that are positive voltage driven!

Macros

- #define **INVERTED_LED_LOGIC** 1

  Flag for calculating duty cycle if LED is common anode.
- #define **CALCULATE_DUTY_CYCLE**(duty, max_duty) (max_duty - duty)

Enumerations

- enum led_state_t { **LED_OFF** = 0 , **LED_ON** = 1 }

  LED state enum.

Functions

- esp_err_t led_driver_init (led_driver_t *led_drv, ledc_timer_bit_t ledc_duty_res, uint32_t ledc_freq)

  Initialize LED driver.
- esp_err_t led_update_duty_cycle (led_driver_t *led_drv, uint16_t duty)

  Update duty cycle of LED.
- esp_err_t led_toggle (led_driver_t *led_drv, led_state_t toggle)

  Toggle LED on/off, and save toggle state to led_drv struct.

4.28.1 Detailed Description

LED driver for ESP32.

4.28.2 Function Documentation

4.28.2.1 led_driver_init()

```
esp_err_t led_driver_init (
            led_driver_t * led_drv,
            ledc_timer_bit_t ledc_duty_res,
            uint32_t ledc_freq )
```

Initialize LED driver.

Parameters

led_drv	Pointer to led_driver_t struct

Returns

ESP_OK on success, ESP_FAIL otherwise

4.28.2.2 led_toggle()

```
esp_err_t led_toggle (
            led_driver_t * led_drv,
            led_state_t toggle )
```

Toggle LED on/off, and save toggle state to led_drv struct.

Parameters

led_drv	Pointer to led_driver_t struct
toggle	Toggle LED on/off

Returns

ESP_OK on success, ESP_FAIL otherwise

4.28.2.3 led_update_duty_cycle()

```
esp_err_t led_update_duty_cycle (
            led_driver_t * led_drv,
            uint16_t duty )
```

Update duty cycle of LED.

Parameters

led_drv	Pointer to led_driver_t struct
duty	Duty cycle in range 0-max_duty

Returns

ESP_OK on success, ESP_FAIL otherwise

4.29 led_driver.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002 #pragma once
00003
00004 #include <stdint.h>
00005
00006 #include "driver/ledc.h"
00007 #include "esp_err.h"
00008 #include "esp_log.h"
00009 #include "stdbool.h"
00010
00017 #define INVERTED_LED_LOGIC 1
00018
00019 #ifdef INVERTED_LED_LOGIC
00020 #define CALCULATE_DUTY_CYCLE(duty, max_duty) (max_duty - duty)
00021 #else
00022 #define CALCULATE_DUTY_CYCLE(duty, max_duty) (duty)
00023 #endif
00024
00028 typedef enum {
00029   LED_OFF = 0,
00030   LED_ON = 1,
00031 } led_state_t;
00032
00042 typedef struct {
00043   ledc_mode_t ledc_mode;
00044   uint8_t led_gpio_num;
00045   uint8_t ledc_channel_num;
00046   uint8_t ledc_timer_num;
00047   uint16_t duty;
00048   uint16_t max_duty;
00049   led_state_t toggle;
00050 } led_driver_t;
00051
00057 esp_err_t led_driver_init(led_driver_t *led_drv, ledc_timer_bit_t ledc_duty_res,
00058                           uint32_t ledc_freq);
00059
00066 esp_err_t led_update_duty_cycle(led_driver_t *led_drv, uint16_t duty);
00067
00075 esp_err_t led_toggle(led_driver_t *led_drv, led_state_t toggle);
```

4.30 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/misc/rgb_led_driver.h File Reference

RGB LED driver for ESP32.

```
#include "led_driver.h"
```

Classes

- struct rgb_led_driver_t

  RGB LED driver struct for ESP32.

Enumerations

- enum rgb_led_driver_table_t { **RED_INDEX** = 0 , **GREEN_INDEX** = 1 , **BLUE_INDEX** = 2 , **MAX_COLOR_INDEX** = 3 }

  RGB LED driver table index enum.

Functions

- esp_err_t rgb_led_driver_init (rgb_led_driver_t *rgb_led_drv, ledc_timer_bit_t ledc_duty_res, uint32_t ledc_freq)

  Initialize RGB LED driver.
- esp_err_t rgb_led_update_duty_cycle (rgb_led_driver_t *rgb_led_drv, uint16_t duty[MAX_COLOR_INDEX])

  Update duty cycle of RGB LED.
- esp_err_t rgb_led_toggle (rgb_led_driver_t *rgb_led_drv, led_state_t toggle)

  Toggle RGB LED.

4.30.1 Detailed Description

RGB LED driver for ESP32.

4.30.2 Function Documentation

4.30.2.1 rgb_led_driver_init()

```
esp_err_t rgb_led_driver_init (
            rgb_led_driver_t * rgb_led_drv,
            ledc_timer_bit_t ledc_duty_res,
            uint32_t ledc_freq )
```

Initialize RGB LED driver.

Parameters

rgb_led_drv	Pointer to rgb_led_driver_t struct
ledc_duty_res	LEDC duty resolution
ledc_freq	LEDC frequency

Returns

ESP_OK on success, ESP_FAIL otherwise

4.30.2.2 rgb_led_toggle()

```
esp_err_t rgb_led_toggle (
            rgb_led_driver_t * rgb_led_drv,
            led_state_t toggle )
```

Toggle RGB LED.

Parameters

rgb_led_drv	Pointer to rgb_led_driver_t struct
toggle	LED_ON or LED_OFF

Returns

ESP_OK on success, ESP_FAIL otherwise

4.30.2.3 rgb_led_update_duty_cycle()

```
esp_err_t rgb_led_update_duty_cycle (
            rgb_led_driver_t * rgb_led_drv,
            uint16_t duty[MAX_COLOR_INDEX] )
```

Update duty cycle of RGB LED.

Parameters

rgb_led_drv	Pointer to rgb_led_driver_t struct
duty	Duty cycle in range 0-max_duty

Returns

ESP_OK on success, ESP_FAIL otherwise

4.31 rgb_led_driver.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002 #pragma once
00003
00004 #include "led_driver.h"
00005
00014 typedef enum {
00015   RED_INDEX = 0,
00016   GREEN_INDEX = 1,
00017   BLUE_INDEX = 2,
00018   MAX_COLOR_INDEX = 3
```

```
00019 } rgb_led_driver_table_t;
00020
00028 typedef struct {
00029   led_driver_t led_drv[MAX_COLOR_INDEX];
00030   uint16_t max_duty;
00031 } rgb_led_driver_t;
00032
00040 esp_err_t rgb_led_driver_init(rgb_led_driver_t *rgb_led_drv,
00041                               ledc_timer_bit_t ledc_duty_res,
00042                               uint32_t ledc_freq);
00043
00050 esp_err_t rgb_led_update_duty_cycle(rgb_led_driver_t *rgb_led_drv,
00051                                    uint16_t duty[MAX_COLOR_INDEX]);
00052
00059 esp_err_t rgb_led_toggle(rgb_led_driver_t *rgb_led_drv, led_state_t toggle);
```

4.32 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_↩ ESP32/components/ssd1306/font8x8_basic.h File Reference

Contains 8x8 font map for unicode points U+0000 - U+007F (basic latin)

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

4.32.1 Detailed Description

Contains 8x8 font map for unicode points U+0000 - U+007F (basic latin)

4.33 font8x8_basic.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #pragma once
00004
00010 #include <stdint.h>
00011
00012 /*
00013    Constant: font8x8_basic_tr
00014    Contains an 90 digree transposed 8x8 font map for unicode points
00015    U+0000 - U+007F (basic latin)
00016
00017    To make it easy to use with SSD1306's GDDRAM mapping and API,
00018    this constant is an 90 degree transposed.
00019    The original version written by Marcel Sondaar is availble at:
00020    https://github.com/dhepper/font8x8/blob/master/font8x8_basic.h
00021
00022    Conversion is done via following procedure:
00023
00024         for (int code = 0; code < 128; code++) {
00025                 uint8_t trans[8];
00026                 for (int w = 0; w < 8; w++) {
00027                         trans[w] = 0x00;
00028                         for (int b = 0; b < 8; b++) {
00029                                 trans[w] |= ((font8x8_basic[code][b] & (1 « w))
00030    » w) « b;
00031                         }
00032                 }
00033
00034                 for (int w = 0; w < 8; w++) {
00035                         if (w == 0) { printf("    { "); }
00036                         printf("0x%.2X", trans[w]);
00037                         if (w < 7) { printf(", "); }
00038                         if (w == 7) { printf(" },   // U+00%.2X (%c)\n", code,
00039    code); }
00040                 }
00041         }
00042 */
```

```
00043
00044 static uint8_t font8x8_basic_tr[128][8] = {
00045     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0000 (nul)
00046     {0x00, 0x04, 0x02, 0xFF, 0x02, 0x04, 0x00, 0x00},  // U+0001 (Up Allow)
00047     {0x00, 0x20, 0x40, 0xFF, 0x40, 0x20, 0x00, 0x00},  // U+0002 (Down Allow)
00048     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0003
00049     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0004
00050     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0005
00051     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0006
00052     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0007
00053     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0008
00054     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0009
00055     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+000A
00056     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+000B
00057     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+000C
00058     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+000D
00059     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+000E
00060     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+000F
00061     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0010
00062     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0011
00063     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0012
00064     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0013
00065     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0014
00066     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0015
00067     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0016
00068     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0017
00069     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0018
00070     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0019
00071     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+001A
00072     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+001B
00073     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+001C
00074     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+001D
00075     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+001E
00076     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+001F
00077     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0020 (space)
00078     {0x00, 0x00, 0x06, 0x5F, 0x5F, 0x06, 0x00, 0x00},  // U+0021 (!)
00079     {0x00, 0x03, 0x03, 0x00, 0x03, 0x03, 0x00, 0x00},  // U+0022 (")
00080     {0x14, 0x7F, 0x7F, 0x14, 0x7F, 0x7F, 0x14, 0x00},  // U+0023 (#)
00081     {0x24, 0x2E, 0x6B, 0x6B, 0x3A, 0x12, 0x00, 0x00},  // U+0024 ($)
00082     {0x46, 0x66, 0x30, 0x18, 0x0C, 0x66, 0x62, 0x00},  // U+0025 (%)
00083     {0x30, 0x7A, 0x4F, 0x5D, 0x37, 0x7A, 0x48, 0x00},  // U+0026 (&)
00084     {0x04, 0x07, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00},  // U+0027 (')
00085     {0x00, 0x1C, 0x3E, 0x63, 0x41, 0x00, 0x00, 0x00},  // U+0028 (()
00086     {0x00, 0x41, 0x63, 0x3E, 0x1C, 0x00, 0x00, 0x00},  // U+0029 ())
00087     {0x08, 0x2A, 0x3E, 0x1C, 0x1C, 0x3E, 0x2A, 0x08},  // U+002A (*)
00088     {0x08, 0x08, 0x3E, 0x3E, 0x08, 0x08, 0x00, 0x00},  // U+002B (+)
00089     {0x00, 0x80, 0xE0, 0x60, 0x00, 0x00, 0x00, 0x00},  // U+002C (,)
00090     {0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x00, 0x00},  // U+002D (-)
00091     {0x00, 0x00, 0x60, 0x60, 0x00, 0x00, 0x00, 0x00},  // U+002E (.)
00092     {0x60, 0x30, 0x18, 0x0C, 0x06, 0x03, 0x01, 0x00},  // U+002F (/)
00093     {0x3E, 0x7F, 0x71, 0x59, 0x4D, 0x7F, 0x3E, 0x00},  // U+0030 (0)
00094     {0x40, 0x42, 0x7F, 0x7F, 0x40, 0x40, 0x00, 0x00},  // U+0031 (1)
00095     {0x62, 0x73, 0x59, 0x49, 0x6F, 0x66, 0x00, 0x00},  // U+0032 (2)
00096     {0x22, 0x63, 0x49, 0x49, 0x7F, 0x36, 0x00, 0x00},  // U+0033 (3)
00097     {0x18, 0x1C, 0x16, 0x53, 0x7F, 0x7F, 0x50, 0x00},  // U+0034 (4)
00098     {0x27, 0x67, 0x45, 0x45, 0x7D, 0x39, 0x00, 0x00},  // U+0035 (5)
00099     {0x3C, 0x7E, 0x4B, 0x49, 0x79, 0x30, 0x00, 0x00},  // U+0036 (6)
00100     {0x03, 0x03, 0x71, 0x79, 0x0F, 0x07, 0x00, 0x00},  // U+0037 (7)
00101     {0x36, 0x7F, 0x49, 0x49, 0x7F, 0x36, 0x00, 0x00},  // U+0038 (8)
00102     {0x06, 0x4F, 0x49, 0x69, 0x3F, 0x1E, 0x00, 0x00},  // U+0039 (9)
00103     {0x00, 0x00, 0x66, 0x66, 0x00, 0x00, 0x00, 0x00},  // U+003A (:)
00104     {0x00, 0x80, 0xE6, 0x66, 0x00, 0x00, 0x00, 0x00},  // U+003B (;)
00105     {0x08, 0x1C, 0x36, 0x63, 0x41, 0x00, 0x00, 0x00},  // U+003C (<)
00106     {0x24, 0x24, 0x24, 0x24, 0x24, 0x24, 0x00, 0x00},  // U+003D (=)
00107     {0x00, 0x41, 0x63, 0x36, 0x1C, 0x08, 0x00, 0x00},  // U+003E (>)
00108     {0x02, 0x03, 0x51, 0x59, 0x0F, 0x06, 0x00, 0x00},  // U+003F (?)
00109     {0x3E, 0x7F, 0x41, 0x5D, 0x5D, 0x1F, 0x1E, 0x00},  // U+0040 (@)
00110     {0x7C, 0x7E, 0x13, 0x13, 0x7E, 0x7C, 0x00, 0x00},  // U+0041 (A)
00111     {0x41, 0x7F, 0x7F, 0x49, 0x49, 0x7F, 0x36, 0x00},  // U+0042 (B)
00112     {0x1C, 0x3E, 0x63, 0x41, 0x41, 0x63, 0x22, 0x00},  // U+0043 (C)
00113     {0x41, 0x7F, 0x7F, 0x41, 0x63, 0x3E, 0x1C, 0x00},  // U+0044 (D)
00114     {0x41, 0x7F, 0x7F, 0x49, 0x5D, 0x41, 0x63, 0x00},  // U+0045 (E)
00115     {0x41, 0x7F, 0x7F, 0x49, 0x1D, 0x01, 0x03, 0x00},  // U+0046 (F)
00116     {0x1C, 0x3E, 0x63, 0x41, 0x51, 0x73, 0x72, 0x00},  // U+0047 (G)
00117     {0x7F, 0x7F, 0x08, 0x08, 0x7F, 0x7F, 0x00, 0x00},  // U+0048 (H)
00118     {0x00, 0x41, 0x7F, 0x7F, 0x41, 0x00, 0x00, 0x00},  // U+0049 (I)
00119     {0x30, 0x70, 0x40, 0x41, 0x7F, 0x3F, 0x01, 0x00},  // U+004A (J)
00120     {0x41, 0x7F, 0x7F, 0x08, 0x1C, 0x77, 0x63, 0x00},  // U+004B (K)
00121     {0x41, 0x7F, 0x7F, 0x41, 0x40, 0x60, 0x70, 0x00},  // U+004C (L)
00122     {0x7F, 0x7F, 0x0E, 0x1C, 0x0E, 0x7F, 0x7F, 0x00},  // U+004D (M)
00123     {0x7F, 0x7F, 0x06, 0x0C, 0x18, 0x7F, 0x7F, 0x00},  // U+004E (N)
00124     {0x1C, 0x3E, 0x63, 0x41, 0x63, 0x3E, 0x1C, 0x00},  // U+004F (O)
00125     {0x41, 0x7F, 0x7F, 0x49, 0x09, 0x0F, 0x06, 0x00},  // U+0050 (P)
00126     {0x1E, 0x3F, 0x21, 0x71, 0x7F, 0x5E, 0x00, 0x00},  // U+0051 (Q)
00127     {0x41, 0x7F, 0x7F, 0x09, 0x19, 0x7F, 0x66, 0x00},  // U+0052 (R)
00128     {0x26, 0x6F, 0x4D, 0x59, 0x73, 0x32, 0x00, 0x00},  // U+0053 (S)
00129     {0x03, 0x41, 0x7F, 0x7F, 0x41, 0x03, 0x00, 0x00},  // U+0054 (T)
```

```
00130     {0x7F, 0x7F, 0x40, 0x40, 0x7F, 0x7F, 0x00, 0x00},  // U+0055 (U)
00131     {0x1F, 0x3F, 0x60, 0x60, 0x3F, 0x1F, 0x00, 0x00},  // U+0056 (V)
00132     {0x7F, 0x7F, 0x30, 0x18, 0x30, 0x7F, 0x7F, 0x00},  // U+0057 (W)
00133     {0x43, 0x67, 0x3C, 0x18, 0x3C, 0x67, 0x43, 0x00},  // U+0058 (X)
00134     {0x07, 0x4F, 0x78, 0x78, 0x4F, 0x07, 0x00, 0x00},  // U+0059 (Y)
00135     {0x47, 0x63, 0x71, 0x59, 0x4D, 0x67, 0x73, 0x00},  // U+005A (Z)
00136     {0x00, 0x7F, 0x7F, 0x41, 0x41, 0x00, 0x00, 0x00},  // U+005B ([)
00137     {0x01, 0x03, 0x06, 0x0C, 0x18, 0x30, 0x60, 0x00},  // U+005C (\)
00138     {0x00, 0x41, 0x41, 0x7F, 0x7F, 0x00, 0x00, 0x00},  // U+005D (])
00139     {0x08, 0x0C, 0x06, 0x03, 0x06, 0x0C, 0x08, 0x00},  // U+005E (^)
00140     {0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80},  // U+005F (_)
00141     {0x00, 0x00, 0x03, 0x07, 0x04, 0x00, 0x00, 0x00},  // U+0060 (`)
00142     {0x20, 0x74, 0x54, 0x54, 0x3C, 0x78, 0x40, 0x00},  // U+0061 (a)
00143     {0x41, 0x7F, 0x3F, 0x48, 0x48, 0x78, 0x30, 0x00},  // U+0062 (b)
00144     {0x38, 0x7C, 0x44, 0x44, 0x6C, 0x28, 0x00, 0x00},  // U+0063 (c)
00145     {0x30, 0x78, 0x48, 0x49, 0x3F, 0x7F, 0x40, 0x00},  // U+0064 (d)
00146     {0x38, 0x7C, 0x54, 0x54, 0x5C, 0x18, 0x00, 0x00},  // U+0065 (e)
00147     {0x48, 0x7E, 0x7F, 0x49, 0x03, 0x02, 0x00, 0x00},  // U+0066 (f)
00148     {0x98, 0xBC, 0xA4, 0xA4, 0xF8, 0x7C, 0x04, 0x00},  // U+0067 (g)
00149     {0x41, 0x7F, 0x7F, 0x08, 0x04, 0x7C, 0x78, 0x00},  // U+0068 (h)
00150     {0x00, 0x44, 0x7D, 0x7D, 0x40, 0x00, 0x00, 0x00},  // U+0069 (i)
00151     {0x60, 0xE0, 0x80, 0x80, 0xFD, 0x7D, 0x00, 0x00},  // U+006A (j)
00152     {0x41, 0x7F, 0x7F, 0x10, 0x38, 0x6C, 0x44, 0x00},  // U+006B (k)
00153     {0x00, 0x41, 0x7F, 0x7F, 0x40, 0x00, 0x00, 0x00},  // U+006C (l)
00154     {0x7C, 0x7C, 0x18, 0x38, 0x1C, 0x7C, 0x78, 0x00},  // U+006D (m)
00155     {0x7C, 0x7C, 0x04, 0x04, 0x7C, 0x78, 0x00, 0x00},  // U+006E (n)
00156     {0x38, 0x7C, 0x44, 0x44, 0x7C, 0x38, 0x00, 0x00},  // U+006F (o)
00157     {0x84, 0xFC, 0xF8, 0xA4, 0x24, 0x3C, 0x18, 0x00},  // U+0070 (p)
00158     {0x18, 0x3C, 0x24, 0xA4, 0xF8, 0xFC, 0x84, 0x00},  // U+0071 (q)
00159     {0x44, 0x7C, 0x78, 0x4C, 0x04, 0x1C, 0x18, 0x00},  // U+0072 (r)
00160     {0x48, 0x5C, 0x54, 0x54, 0x74, 0x24, 0x00, 0x00},  // U+0073 (s)
00161     {0x00, 0x04, 0x3E, 0x7F, 0x44, 0x24, 0x00, 0x00},  // U+0074 (t)
00162     {0x3C, 0x7C, 0x40, 0x40, 0x3C, 0x7C, 0x40, 0x00},  // U+0075 (u)
00163     {0x1C, 0x3C, 0x60, 0x60, 0x3C, 0x1C, 0x00, 0x00},  // U+0076 (v)
00164     {0x3C, 0x7C, 0x70, 0x38, 0x70, 0x7C, 0x3C, 0x00},  // U+0077 (w)
00165     {0x44, 0x6C, 0x38, 0x10, 0x38, 0x6C, 0x44, 0x00},  // U+0078 (x)
00166     {0x9C, 0xBC, 0xA0, 0xA0, 0xFC, 0x7C, 0x00, 0x00},  // U+0079 (y)
00167     {0x4C, 0x64, 0x74, 0x5C, 0x4C, 0x64, 0x00, 0x00},  // U+007A (z)
00168     {0x08, 0x08, 0x3E, 0x77, 0x41, 0x41, 0x00, 0x00},  // U+007B ({)
00169     {0x00, 0x00, 0x00, 0x77, 0x77, 0x00, 0x00, 0x00},  // U+007C (|)
00170     {0x41, 0x41, 0x77, 0x3E, 0x08, 0x08, 0x00, 0x00},  // U+007D (})
00171     {0x02, 0x03, 0x01, 0x03, 0x02, 0x03, 0x01, 0x00},  // U+007E (~)
00172     {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}   // U+007F
00173 };
```

4.34 /Users/krzysztofgliwinski/PoliWRocket/ROSALIA/Mainboard_ESP32/components/ssd1306/ssd1306.h File Reference

SSD1306 OLED display driver through I2C.

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <string.h>
#include "font8x8_basic.h"
```

Classes

- struct PAGE_t

  SSD1306 display page structure.

- struct ssd1306_t

  SSD1306 display structure.

Macros

- #define **OLED_CONTROL_BYTE_CMD_SINGLE** 0x80
- #define **OLED_CONTROL_BYTE_CMD_STREAM** 0x00
- #define **OLED_CONTROL_BYTE_DATA_SINGLE** 0xC0
- #define **OLED_CONTROL_BYTE_DATA_STREAM** 0x40
- #define **OLED_CMD_SET_CONTRAST** 0x81
- #define **OLED_CMD_DISPLAY_RAM** 0xA4
- #define **OLED_CMD_DISPLAY_ALLON** 0xA5
- #define **OLED_CMD_DISPLAY_NORMAL** 0xA6
- #define **OLED_CMD_DISPLAY_INVERTED** 0xA7
- #define **OLED_CMD_DISPLAY_OFF** 0xAE
- #define **OLED_CMD_DISPLAY_ON** 0xAF
- #define **OLED_CMD_SET_MEMORY_ADDR_MODE** 0x20
- #define **OLED_CMD_SET_HORI_ADDR_MODE** 0x00
- #define **OLED_CMD_SET_VERT_ADDR_MODE** 0x01
- #define **OLED_CMD_SET_PAGE_ADDR_MODE** 0x02
- #define **OLED_CMD_SET_COLUMN_RANGE** 0x21
- #define **OLED_CMD_SET_PAGE_RANGE** 0x22
- #define **OLED_CMD_SET_DISPLAY_START_LINE** 0x40
- #define **OLED_CMD_SET_SEGMENT_REMAP_0** 0xA0
- #define **OLED_CMD_SET_SEGMENT_REMAP_1** 0xA1
- #define **OLED_CMD_SET_MUX_RATIO** 0xA8
- #define **OLED_CMD_SET_COM_SCAN_MODE** 0xC8
- #define **OLED_CMD_SET_DISPLAY_OFFSET** 0xD3
- #define **OLED_CMD_SET_COM_PIN_MAP** 0xDA
- #define **OLED_CMD_NOP** 0xE3
- #define **OLED_CMD_SET_DISPLAY_CLK_DIV** 0xD5
- #define **OLED_CMD_SET_PRECHARGE** 0xD9
- #define **OLED_CMD_SET_VCOMH_DESELCT** 0xDB
- #define **OLED_CMD_SET_CHARGE_PUMP** 0x8D
- #define **OLED_CMD_HORIZONTAL_RIGHT** 0x26
- #define **OLED_CMD_HORIZONTAL_LEFT** 0x27
- #define **OLED_CMD_CONTINUOUS_SCROLL** 0x29
- #define **OLED_CMD_DEACTIVE_SCROLL** 0x2E
- #define **OLED_CMD_ACTIVE_SCROLL** 0x2F
- #define **OLED_CMD_VERTICAL** 0xA3
- #define **I2CAddress** 0x3C
- #define **OLED_BUFFER_SIZE** 128

Typedefs

- typedef void * **ssd1306_i2c_cmd_handle_t**
- typedef bool(* **ssd1306_i2c_master_write_byte**) (ssd1306_i2c_cmd_handle_t cmd, uint8_t _data, bool _ack_en)
- typedef bool(* **ssd1306_i2c_master_write**) (ssd1306_i2c_cmd_handle_t cmd, const uint8_t *_data, size_t _data_len, bool _ack_en)
- typedef bool(* **ssd1306_i2c_master_start**) (ssd1306_i2c_cmd_handle_t cmd)
- typedef bool(* **ssd1306_i2c_master_stop**) ()
- typedef bool(* **ssd1306_i2c_master_cmd_begin**) (ssd1306_i2c_cmd_handle_t cmd, uint16_t ticks_to_wait)
- typedef ssd1306_i2c_cmd_handle_t(* **ssd1306_i2c_cmd_link_create**) ()
- typedef void(* **ssd1306_i2c_cmd_link_delete**) (ssd1306_i2c_cmd_handle_t cmd)
- typedef void(* **ssd1306_delay**) (size_t _ms)
- typedef void(* **ssd1306_log**) (const ssd1306_log_level_t level, const char *tag, char *info)

Enumerations

- enum ssd1306_scroll_type_t { **SCROLL_RIGHT** = 1 , **SCROLL_LEFT** = 2 , **SCROLL_DOWN** = 3 , **SCROLL_UP** = 4 , **SCROLL_STOP** = 5 }

  enum for scroll type
- enum ssd1306_log_level_t { **SSD1306_NONE** , **SSD1306_ERROR** , **SSD1306_INFO** , **SSD1306_DEBUG** }

  enum for log level

Functions

- void ssd1306_init (ssd1306_t *ssd, uint8_t width, uint8_t height)

  Initialize SSD1306 display.
- void ssd1306_show_buffer (ssd1306_t *ssd)

  Display whole image from buffer.
- void ssd1306_set_buffer (ssd1306_t *ssd, uint8_t *buffer)

  Set buffer for display from array to pages.
- void ssd1306_get_buffer (ssd1306_t *ssd, uint8_t *buffer)

  Get buffer from display and save to buffer array.
- void ssd1306_display_image (ssd1306_t *ssd, int page, int seg, uint8_t *images, uint8_t width)

  Display image from buffer.
- void ssd1306_display_text (ssd1306_t *ssd, int page, char *text, int text_len, bool invert)

  Display text on display.
- void ssd1306_display_text_x3 (ssd1306_t *ssd, int page, char *text, int text_len, bool invert)

  Display text on display with x3 size.
- void ssd1306_clear_screen (ssd1306_t *ssd, bool invert)

  Clear the whole screen.
- void ssd1306_clear_line (ssd1306_t *ssd, int page, bool invert)

  Clear the whole screen.
- void ssd1306_set_contrast (ssd1306_t *ssd, int contrast)

  Set contrast of display.
- void ssd1306_software_scroll (ssd1306_t *ssd, int start, int end)

  Perform software scroll of pixels.
- void ssd1306_scroll_text (ssd1306_t *ssd, char *text, int text_len, bool invert)

  Perform hardware scroll of text.
- void ssd1306_scroll_clear (ssd1306_t *ssd)

  Clear image with scroll.
- void ssd1306_hardware_scroll (ssd1306_t *ssd, ssd1306_scroll_type_t scroll)

  Perform hardware scroll of pixels.
- void ssd1306_wrap_arround (ssd1306_t *ssd, ssd1306_scroll_type_t scroll, int start, int end, int8_t delay)

  Wrap around image through scroll.
- void ssd1306_show_bitmap (ssd1306_t *ssd, int xpos, int ypos, uint8_t *bitmap, uint8_t width, uint8_t height, bool invert)

  Show bitmap on display.
- void _ssd1306_pixel (ssd1306_t *ssd, int xpos, int ypos, bool invert)

  Save pixel to internal buffer, without displaying.
- void _ssd1306_line (ssd1306_t *ssd, int x1, int y1, int x2, int y2, bool invert)

  Save a line to internal buffer, without displaying.
- void ssd1306_invert (uint8_t *buf, size_t blen)

  Invert a buffer.
- void ssd1306_flip (uint8_t *buf, size_t blen)

Flip a buffer.

- uint8_t ssd1306_copy_bit (uint8_t src, int srcBits, uint8_t dst, int dstBits)

  Copy a bit from one byte to another.
- uint8_t ssd1306_rotate_byte (uint8_t ch1)

  Rotate a byte.
- void ssd1306_fadeout (ssd1306_t *ssd)

  Perform a fade out effect.
- void ssd1306_i2c_init (ssd1306_t *ssd, uint8_t width, uint8_t height)

  Init of the screen through i2c.
- void ssd1306_i2c_display_image (ssd1306_t *ssd, int page, int seg, uint8_t *images, uint8_t width)

  Display an image through i2c.
- void ssd1306_i2c_set_contrast (ssd1306_t *ssd, int contrast)

  Set the display contrast through i2c.
- void ssd1306_i2c_hardware_scroll (ssd1306_t *ssd, ssd1306_scroll_type_t scroll)

  Perform hardware scroll of pixels through i2c.

4.34.1 Detailed Description

SSD1306 OLED display driver through I2C.

4.34.2 Function Documentation

4.34.2.1 _ssd1306_line()

```
void _ssd1306_line (
            ssd1306_t * ssd,
            int x1,
            int y1,
            int x2,
            int y2,
            bool invert )
```

Save a line to internal buffer, without displaying.

Parameters

in	*ssd*	SSD1306 display structure
in	*x1*	X1 position
in	*y1*	Y1 position
in	*x2*	X2 position
in	*y2*	Y2 position
in	*invert*	Invert line

4.34.2.2 _ssd1306_pixel()

```
void _ssd1306_pixel (
            ssd1306_t * ssd,
```

```
            int xpos,
            int ypos,
            bool invert )
```

Save pixel to internal buffer, without displaying.

Parameters

in	*ssd*	SSD1306 display structure
in	*xpos*	X position
in	*ypos*	Y position
in	*invert*	Invert pixel

4.34.2.3 ssd1306_clear_line()

```
void ssd1306_clear_line (
            ssd1306_t * ssd,
            int page,
            bool invert )
```

Clear the whole screen.

Parameters

in	*ssd*	SSD1306 display structure
in	*page*	Page to clear
in	*invert*	Invert screen

4.34.2.4 ssd1306_clear_screen()

```
void ssd1306_clear_screen (
            ssd1306_t * ssd,
            bool invert )
```

Clear the whole screen.

Parameters

in	*ssd*	SSD1306 display structure
in	*invert*	Invert screen

4.34.2.5 ssd1306_copy_bit()

```
uint8_t ssd1306_copy_bit (
            uint8_t src,
            int srcBits,
            uint8_t dst,
            int dstBits )
```

Copy a bit from one byte to another.

Parameters

in	*src*	Source byte
in	*srcBits*	Source bit position
in	*dst*	Destination byte
in	*dstBits*	Destination bit position

4.34.2.6 ssd1306_display_image()

```
void ssd1306_display_image (
            ssd1306_t * ssd,
            int page,
            int seg,
            uint8_t * images,
            uint8_t width )
```

Display image from buffer.

Parameters

in	*ssd*	SSD1306 display structure
in	*page*	Page to display
in	*seg*	Segment to display
in	*images*	Image to display
in	*width*	Image width

4.34.2.7 ssd1306_display_text()

```
void ssd1306_display_text (
            ssd1306_t * ssd,
            int page,
            char * text,
            int text_len,
            bool invert )
```

Display text on display.

Parameters

in	*ssd*	SSD1306 display structure
in	*page*	Page to display
in	*text*	Text to display
in	*text_len*	Text length
in	*invert*	Invert text

4.34.2.8 ssd1306_display_text_x3()

```
void ssd1306_display_text_x3 (
```

```
            ssd1306_t * ssd,
            int page,
            char * text,
            int text_len,
            bool invert )
```

Display text on display with x3 size.

Parameters

in	*ssd*	SSD1306 display structure
in	*page*	Page to display
in	*text*	Text to display
in	*text_len*	Text length
in	*invert*	Invert text

4.34.2.9 ssd1306_fadeout()

```
void ssd1306_fadeout (
            ssd1306_t * ssd )
```

Perform a fade out effect.

Parameters

in	*ssd*	SSD1306 display structure

4.34.2.10 ssd1306_flip()

```
void ssd1306_flip (
            uint8_t * buf,
            size_t blen )
```

Flip a buffer.

Parameters

in	*buf*	Buffer to flip, after this operation buffer will be flipped
in	*blen*	Buffer length

4.34.2.11 ssd1306_get_buffer()

```
void ssd1306_get_buffer (
            ssd1306_t * ssd,
            uint8_t * buffer )
```

Get buffer from display and save to buffer array.

Parameters

in	*ssd*	SSD1306 display structure
out	*buffer*	Buffer to get

4.34.2.12 ssd1306_hardware_scroll()

```
void ssd1306_hardware_scroll (
            ssd1306_t * ssd,
            ssd1306_scroll_type_t scroll )
```

Perform hardware scroll of pixels.

Parameters

in	*ssd*	SSD1306 display structure
in	*scroll*	Scroll type

4.34.2.13 ssd1306_i2c_display_image()

```
void ssd1306_i2c_display_image (
            ssd1306_t * ssd,
            int page,
            int seg,
            uint8_t * images,
            uint8_t width )
```

Display an image through i2c.

Parameters

in	*ssd*	SSD1306 display structure
in	*page*	Page to display
in	*seg*	Segment to display
in	*images*	Image to display
in	*width*	Image width

Note

This function is defined in ssd1306_i2c.c

4.34.2.14 ssd1306_i2c_hardware_scroll()

```
void ssd1306_i2c_hardware_scroll (
            ssd1306_t * ssd,
            ssd1306_scroll_type_t scroll )
```

Perform hardware scroll of pixels through i2c.

Parameters

in	*ssd*	SSD1306 display structure
in	*scroll*	Scroll type

Note

This function is defined in ssd1306_i2c.c

4.34.2.15 ssd1306_i2c_init()

```
void ssd1306_i2c_init (
            ssd1306_t * ssd,
            uint8_t width,
            uint8_t height )
```

Init of the screen through i2c.

Parameters

in	*ssd*	SSD1306 display structure
in	*width*	Display width
in	*height*	Display height

Note

This function is defined in ssd1306_i2c.c

4.34.2.16 ssd1306_i2c_set_contrast()

```
void ssd1306_i2c_set_contrast (
            ssd1306_t * ssd,
            int contrast )
```

Set the display contrast through i2c.

Parameters

in	*ssd*	SSD1306 display structure
in	*contrast*	Contrast to set

Note

This function is defined in ssd1306_i2c.c

4.34.2.17 ssd1306_init()

```
void ssd1306_init (
```

```
            ssd1306_t * ssd,
            uint8_t width,
            uint8_t height )
```

Initialize SSD1306 display.

Parameters

in	*ssd*	SSD1306 display structure
in	*width*	Display width
in	*height*	Display height

4.34.2.18 ssd1306_invert()

```
void ssd1306_invert (
            uint8_t * buf,
            size_t blen )
```

Invert a buffer.

Parameters

in	*buf*	Buffer to invert, after this operation buffer will be inverted
in	*blen*	Buffer length

4.34.2.19 ssd1306_rotate_byte()

```
uint8_t ssd1306_rotate_byte (
            uint8_t ch1 )
```

Rotate a byte.

Parameters

in	*ch1*	Byte to rotate

Returns

Rotated byte

4.34.2.20 ssd1306_scroll_clear()

```
void ssd1306_scroll_clear (
            ssd1306_t * ssd )
```

Clear image with scroll.

Parameters

in	*ssd*	SSD1306 display structure

4.34.2.21 ssd1306_scroll_text()

```
void ssd1306_scroll_text (
            ssd1306_t * ssd,
            char * text,
            int text_len,
            bool invert )
```

Perform hardware scroll of text.

Parameters

in	*ssd*	SSD1306 display structure
in	*text*	Text to scroll
in	*text_len*	Text length
in	*invert*	Invert text

4.34.2.22 ssd1306_set_buffer()

```
void ssd1306_set_buffer (
            ssd1306_t * ssd,
            uint8_t * buffer )
```

Set buffer for display from array to pages.

Parameters

in	*ssd*	SSD1306 display structure
in	*buffer*	Buffer to set

4.34.2.23 ssd1306_set_contrast()

```
void ssd1306_set_contrast (
            ssd1306_t * ssd,
            int contrast )
```

Set contrast of display.

Parameters

in	*ssd*	SSD1306 display structure
in	*contrast*	Contrast value

4.34.2.24 ssd1306_show_bitmap()

```
void ssd1306_show_bitmap (
            ssd1306_t * ssd,
            int xpos,
            int ypos,
            uint8_t * bitmap,
            uint8_t width,
            uint8_t height,
            bool invert )
```

Show bitmap on display.

Parameters

in	*ssd*	SSD1306 display structure
in	*xpos*	X position
in	*ypos*	Y position
in	*bitmap*	Bitmap to display
in	*width*	Bitmap width
in	*height*	Bitmap height
in	*invert*	Invert bitmap

4.34.2.25 ssd1306_show_buffer()

```
void ssd1306_show_buffer (
            ssd1306_t * ssd )
```

Display whole image from buffer.

Parameters

in	*ssd*	SSD1306 display structure

4.34.2.26 ssd1306_software_scroll()

```
void ssd1306_software_scroll (
            ssd1306_t * ssd,
            int start,
            int end )
```

Perform software scroll of pixels.

Parameters

in	*ssd*	SSD1306 display structure
in	*start*	Start page
in	*end*	End page

4.34.2.27 ssd1306_wrap_arround()

```
void ssd1306_wrap_arround (
            ssd1306_t * ssd,
            ssd1306_scroll_type_t scroll,
            int start,
            int end,
            int8_t delay )
```

Wrap around image through scroll.

Parameters

in	*ssd*	SSD1306 display structure
in	*scroll*	Scroll type
in	*start*	Start page
in	*end*	End page
in	*delay*	Delay between scroll

4.35 ssd1306.h

Go to the documentation of this file.

```
00001 // Copyright 2023 PWr in Space, Krzysztof Gliwiński
00002
00003 #pragma once
00004
00005 #include <stdbool.h>
00006 #include <stddef.h>
00007 #include <stdint.h>
00008 #include <string.h>
00009
00010 #include "font8x8_basic.h"
00016 // Following definitions are bollowed from
00017 // http://robotcantalk.blogspot.com/2015/03/interfacing-arduino-with-ssd1306-driven.html
00018
00019 /* Control byte for i2c
00020 Co : bit 8 : Continuation Bit
00021  * 1 = no-continuation (only one byte to follow)
00022  * 0 = the controller should expect a stream of bytes.
00023 D/C# : bit 7 : Data/Command Select bit
00024  * 1 = the next byte or byte stream will be Data.
00025  * 0 = a Command byte or byte stream will be coming up next.
00026  Bits 6-0 will be all zeros.
00027 Usage:
00028 0x80 : Single Command byte
00029 0x00 : Command Stream
00030 0xC0 : Single Data byte
00031 0x40 : Data Stream
00032 */
00033 #define OLED_CONTROL_BYTE_CMD_SINGLE 0x80
00034 #define OLED_CONTROL_BYTE_CMD_STREAM 0x00
00035 #define OLED_CONTROL_BYTE_DATA_SINGLE 0xC0
00036 #define OLED_CONTROL_BYTE_DATA_STREAM 0x40
00037
00038 // Fundamental commands (pg.28)
00039 #define OLED_CMD_SET_CONTRAST 0x81  // follow with 0x7F
00040 #define OLED_CMD_DISPLAY_RAM 0xA4
00041 #define OLED_CMD_DISPLAY_ALLON 0xA5
00042 #define OLED_CMD_DISPLAY_NORMAL 0xA6
00043 #define OLED_CMD_DISPLAY_INVERTED 0xA7
00044 #define OLED_CMD_DISPLAY_OFF 0xAE
00045 #define OLED_CMD_DISPLAY_ON 0xAF
00046
00047 // Addressing Command Table (pg.30)
00048 #define OLED_CMD_SET_MEMORY_ADDR_MODE 0x20
00049 #define OLED_CMD_SET_HORI_ADDR_MODE 0x00  // Horizontal Addressing Mode
00050 #define OLED_CMD_SET_VERT_ADDR_MODE 0x01  // Vertical Addressing Mode
00051 #define OLED_CMD_SET_PAGE_ADDR_MODE 0x02  // Page Addressing Mode
```

```
00052 #define OLED_CMD_SET_COLUMN_RANGE \
00053   0x21  // can be used only in HORZ/VERT mode - follow with 0x00 and 0x7F =
00054         // COL127
00055 #define OLED_CMD_SET_PAGE_RANGE \
00056   0x22  // can be used only in HORZ/VERT mode - follow with 0x00 and 0x07 =
00057         // PAGE7
00058
00059 // Hardware Config (pg.31)
00060 #define OLED_CMD_SET_DISPLAY_START_LINE 0x40
00061 #define OLED_CMD_SET_SEGMENT_REMAP_0 0xA0
00062 #define OLED_CMD_SET_SEGMENT_REMAP_1 0xA1
00063 #define OLED_CMD_SET_MUX_RATIO 0xA8  // follow with 0x3F = 64 MUX
00064 #define OLED_CMD_SET_COM_SCAN_MODE 0xC8
00065 #define OLED_CMD_SET_DISPLAY_OFFSET 0xD3  // follow with 0x00
00066 #define OLED_CMD_SET_COM_PIN_MAP 0xDA     // follow with 0x12
00067 #define OLED_CMD_NOP 0xE3                 // NOP
00068
00069 // Timing and Driving Scheme (pg.32)
00070 #define OLED_CMD_SET_DISPLAY_CLK_DIV 0xD5  // follow with 0x80
00071 #define OLED_CMD_SET_PRECHARGE 0xD9        // follow with 0xF1
00072 #define OLED_CMD_SET_VCOMH_DESELCT 0xDB    // follow with 0x30
00073
00074 // Charge Pump (pg.62)
00075 #define OLED_CMD_SET_CHARGE_PUMP 0x8D  // follow with 0x14
00076
00077 // Scrolling Commands
00078 #define OLED_CMD_HORIZONTAL_RIGHT 0x26
00079 #define OLED_CMD_HORIZONTAL_LEFT 0x27
00080 #define OLED_CMD_CONTINUOUS_SCROLL 0x29
00081 #define OLED_CMD_DEACTIVE_SCROLL 0x2E
00082 #define OLED_CMD_ACTIVE_SCROLL 0x2F
00083 #define OLED_CMD_VERTICAL 0xA3
00084
00085 // I2C Address
00086 #define I2CAddress 0x3C  // TODO(Glibus): Move this to kconfig_projbuild
00087
00088 // Internal buffer size
00089 #define OLED_BUFFER_SIZE 128
00090
00094 typedef enum {
00095   SCROLL_RIGHT = 1,
00096   SCROLL_LEFT = 2,
00097   SCROLL_DOWN = 3,
00098   SCROLL_UP = 4,
00099   SCROLL_STOP = 5
00100 } ssd1306_scroll_type_t;
00101
00105 typedef enum {
00106   SSD1306_NONE,
00107   SSD1306_ERROR,
00108   SSD1306_INFO,
00109   SSD1306_DEBUG,
00110 } ssd1306_log_level_t;
00111
00112 typedef void* ssd1306_i2c_cmd_handle_t;
00113
00114 typedef bool (*ssd1306_i2c_master_write_byte)(ssd1306_i2c_cmd_handle_t cmd,
00115                                               uint8_t _data, bool _ack_en);
00116 typedef bool (*ssd1306_i2c_master_write)(ssd1306_i2c_cmd_handle_t cmd,
00117                                          const uint8_t* _data, size_t _data_len,
00118                                          bool _ack_en);
00119 typedef bool (*ssd1306_i2c_master_start)(ssd1306_i2c_cmd_handle_t cmd);
00120 typedef bool (*ssd1306_i2c_master_stop)();
00121 typedef bool (*ssd1306_i2c_master_cmd_begin)(ssd1306_i2c_cmd_handle_t cmd,
00122                                              uint16_t ticks_to_wait);
00123 typedef ssd1306_i2c_cmd_handle_t (*ssd1306_i2c_cmd_link_create)();
00124 typedef void (*ssd1306_i2c_cmd_link_delete)(ssd1306_i2c_cmd_handle_t cmd);
00125 typedef void (*ssd1306_delay)(size_t _ms);
00126 typedef void (*ssd1306_log)(const ssd1306_log_level_t level, const char* tag,
00127                             char* info);
00128
00132 typedef struct {
00133   uint8_t segments[OLED_BUFFER_SIZE];
00134 } PAGE_t;
00135
00139 typedef struct {
00140   ssd1306_i2c_master_write_byte _i2c_master_write_byte;
00141   ssd1306_i2c_master_write _i2c_master_write;
00142   ssd1306_i2c_master_start _i2c_master_start;
00143   ssd1306_i2c_master_stop _i2c_master_stop;
00144   ssd1306_i2c_master_cmd_begin _i2c_master_cmd_begin;
00145   ssd1306_i2c_cmd_link_create _i2c_cmd_link_create;
00146   ssd1306_i2c_cmd_link_delete _i2c_cmd_link_delete;
00147   ssd1306_delay _delay;
00148   ssd1306_log _log;
00149   uint8_t i2c_master_write_flag;
00150   uint8_t width;
```

```
00151   uint8_t height;
00152   int pages;
00153   bool scroll_enable;
00154   int scroll_start;
00155   int scroll_end;
00156   int scroll_direction;
00157   PAGE_t screen_pages[8];
00158   bool flip;
00159   uint8_t i2c_address;
00160 } ssd1306_t;
00161
00168 void ssd1306_init(ssd1306_t* ssd, uint8_t width, uint8_t height);
00169
00174 void ssd1306_show_buffer(ssd1306_t* ssd);
00175
00181 void ssd1306_set_buffer(ssd1306_t* ssd, uint8_t* buffer);
00182
00188 void ssd1306_get_buffer(ssd1306_t* ssd, uint8_t* buffer);
00189
00198 void ssd1306_display_image(ssd1306_t* ssd, int page, int seg, uint8_t* images,
00199                            uint8_t width);
00200
00209 void ssd1306_display_text(ssd1306_t* ssd, int page, char* text, int text_len,
00210                           bool invert);
00211
00220 void ssd1306_display_text_x3(ssd1306_t* ssd, int page, char* text, int text_len,
00221                              bool invert);
00222
00228 void ssd1306_clear_screen(ssd1306_t* ssd, bool invert);
00229
00236 void ssd1306_clear_line(ssd1306_t* ssd, int page, bool invert);
00237
00243 void ssd1306_set_contrast(ssd1306_t* ssd, int contrast);
00244
00251 void ssd1306_software_scroll(ssd1306_t* ssd, int start, int end);
00252
00260 void ssd1306_scroll_text(ssd1306_t* ssd, char* text, int text_len, bool invert);
00261
00266 void ssd1306_scroll_clear(ssd1306_t* ssd);
00267
00273 void ssd1306_hardware_scroll(ssd1306_t* ssd, ssd1306_scroll_type_t scroll);
00274
00283 void ssd1306_wrap_arround(ssd1306_t* ssd, ssd1306_scroll_type_t scroll,
00284                           int start, int end, int8_t delay);
00295 void ssd1306_show_bitmap(ssd1306_t* ssd, int xpos, int ypos, uint8_t* bitmap,
00296                          uint8_t width, uint8_t height, bool invert);
00297
00305 void _ssd1306_pixel(ssd1306_t* ssd, int xpos, int ypos, bool invert);
00315 void _ssd1306_line(ssd1306_t* ssd, int x1, int y1, int x2, int y2, bool invert);
00321 void ssd1306_invert(uint8_t* buf, size_t blen);
00327 void ssd1306_flip(uint8_t* buf, size_t blen);
00335 uint8_t ssd1306_copy_bit(uint8_t src, int srcBits, uint8_t dst, int dstBits);
00341 uint8_t ssd1306_rotate_byte(uint8_t ch1);
00346 void ssd1306_fadeout(ssd1306_t* ssd);
00347
00355 void ssd1306_i2c_init(ssd1306_t* ssd, uint8_t width, uint8_t height);
00356
00366 void ssd1306_i2c_display_image(ssd1306_t* ssd, int page, int seg,
00367                                uint8_t* images, uint8_t width);
00368
00375 void ssd1306_i2c_set_contrast(ssd1306_t* ssd, int contrast);
00376
00383 void ssd1306_i2c_hardware_scroll(ssd1306_t* ssd, ssd1306_scroll_type_t scroll);
```

Index