SmartHomeLED

Generated by Doxygen 1.8.17

Chapter 1

SmartLed

Repository for Smart LED Control System Senior Design Project

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 channel Struct Reference

Public Attributes

- char name [250]
- uint16_t r
- uint16_t g
- uint16_t b
- uint8_t brightness

4.1.1 Member Data Documentation

4.1.1.1 b

```
uint16_t channel::b
```

4.1.1.2 brightness

```
uint8_t channel::brightness
```

4.1.1.3 g

```
uint16_t channel::g
```

4.1.1.4 name

```
char channel::name[250]
```

4.1.1.5 r

```
uint16_t channel::r
```

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

- src/led.cpp

4.2 gatts_profile_inst Struct Reference

Public Attributes

- esp_gatts_cb_t gatts_cb
- uint16_t gatts_if
- uint16_t app_id
- 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

4.2.1 Member Data Documentation

4.2.1.1 app_id

```
uint16_t gatts_profile_inst::app_id
```

4.2.1.2 char_handle

```
uint16_t gatts_profile_inst::char_handle
```

4.2.1.3 char_uuid

```
esp_bt_uuid_t gatts_profile_inst::char_uuid
```

4.2.1.4 conn_id

```
uint16_t gatts_profile_inst::conn_id
```

4.2.1.5 descr_handle

```
uint16_t gatts_profile_inst::descr_handle
```

4.2.1.6 descr_uuid

```
esp_bt_uuid_t gatts_profile_inst::descr_uuid
```

4.2.1.7 gatts_cb

```
esp_gatts_cb_t gatts_profile_inst::gatts_cb
```

4.2.1.8 gatts_if

```
uint16_t gatts_profile_inst::gatts_if
```

4.2.1.9 perm

```
esp_gatt_perm_t gatts_profile_inst::perm
```

4.2.1.10 property

```
esp_gatt_char_prop_t gatts_profile_inst::property
```

4.2.1.11 service_handle

```
uint16_t gatts_profile_inst::service_handle
```

4.2.1.12 service_id

```
esp_gatt_srvc_id_t gatts_profile_inst::service_id
```

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

- src/bleSL.cpp

4.3 Node Struct Reference

Nodes of a singly linked list data structure.

```
#include <schedule_object.h>
```

Public Attributes

- schedule_object schedule
- struct Node * next

4.3.1 Detailed Description

Nodes of a singly linked list data structure.

4.3.2 Member Data Documentation

4.3.2.1 next

```
struct Node* Node::next
```

Pointer to the next node in the linked list data structure

4.3.2.2 schedule

```
schedule_object Node::schedule
```

the schedule associated with this node

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

- include/schedule_object.h

4.4 prepare_type_env_t Struct Reference

Public Attributes

- uint8_t * prepare_buf
- int prepare_len

4.4.1 Member Data Documentation

4.4.1.1 prepare_buf

```
uint8_t* prepare_type_env_t::prepare_buf
```

4.4.1.2 prepare_len

```
int prepare_type_env_t::prepare_len
```

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

- src/bleSL.cpp

4.5 Schedule_Object Struct Reference

data object that specified parameters a schedule can contain

```
#include <schedule_object.h>
```

Public Attributes

- uint8_t ID
- char name [250]
- uint8_t enabled
- uint32_t start
- uint32_t duration
- uint8_t repeat_mask
- uint32_t repeat_time
- uint8_t dawn
- uint8_t dusk
- uint8_t isRGB
- uint8_t brightness
- uint8_t r
- uint8_t g
- uint8_t b

4.5.1 Detailed Description

data object that specified parameters a schedule can contain

4.5.2 Member Data Documentation

4.5.2.1 b

```
uint8_t Schedule_Object::b
```

The blue portion of the color. Values from 0-255.

4.5.2.2 brightness

```
uint8_t Schedule_Object::brightness
```

The brightness of the LED. 0-255 with 0 being off, and 255 being maximum brightness.

4.5.2.3 dawn

```
uint8_t Schedule_Object::dawn
```

Whether the schedule runs at dawn. Not compatable with repeat_time

4.5.2.4 duration

```
uint32_t Schedule_Object::duration
```

The duration, in seconds, of the schedule. If set to the max value the schedule will run indefinitely.

4.5.2.5 dusk

```
uint8_t Schedule_Object::dusk
```

Whether the schedule runs at dusk. Not compatable with repeat_time.

4.5.2.6 enabled

```
uint8_t Schedule_Object::enabled
```

The enabled status of the schedule. If 0, the schedule does not run. If not 0 the schedule actively runs according to other parameters.

4.5.2.7 g

```
uint8_t Schedule_Object::g
```

The green portion of the color. Values from 0-255.

4.5.2.8 ID

```
uint8_t Schedule_Object::ID
```

The ID for the schedule. Each schedule in a list should have its own unique ID.

4.5.2.9 isRGB

```
uint8_t Schedule_Object::isRGB
```

The type of LED associated. If this field is 0, the LED is not an RGB LED and only the brightness field is used in controlling the LED. If isRGB is not 0, the brightness, R, G, and B fields control the LED

4.5.2.10 name

```
char Schedule_Object::name[250]
```

The name of the schedule. Each schedule in a list should have its own unique name.

4.5.2.11 r

```
uint8_t Schedule_Object::r
```

The red portion of the color. Values from 0-255.

4.5.2.12 repeat_mask

```
uint8_t Schedule_Object::repeat_mask
```

A bit mask for which days the schedule runs on. If repeat_time is set this field is not used. The MSB bit is not used and following bits represent the days. The 8 bits represent [Unused, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday].

4.5.2.13 repeat_time

```
uint32_t Schedule_Object::repeat_time
```

An alternative way to have schedules repeat instead of by day. This field represents the time, in seconds, that the schedule repeats. The next start time is calculated by adding the current start time plus the repeat_time. Settings this field to 5 seconds means that the schedule's start is triggered every 5 seconds. If this field is enabled then repeat_mask and dawn/dusk will not work.

4.5.2.14 start

```
uint32_t Schedule_Object::start
```

The unix timestamp of the start time. If a schedule repeats, this field will update to the next start time as the schedule runs.

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

- include/schedule_object.h

Chapter 5

File Documentation

5.1 include/ArduinoJson-v6.14.1.h File Reference

5.2 include/bleSL.h File Reference

Functions

- void Init_Bluetooth (void)
  Initializes GATT Server.

5.2.1 Detailed Description

Description: Function declaration to initialize BLE GATT Server.

Author

: Hunaid Puri

Date

last modified: 4/26/2020

5.2.2 Function Documentation

5.2.2.1 Init_Bluetooth()

```
void Init_Bluetooth (
            void  )
```

Initializes GATT Server.

5.3 include/dawndusk.h File Reference

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

Macros

- #define radius 6378

  Size of radius of Earth, used in dawn/dusk calculation.
- #define pi 3.1415926

  Value of pi, used in dawn/dusk calculations.
- #define planeDist 23.45

  Distance between x-y plane and ecliptic plane, used in dawn/dusk calculations.
- #define distSun 149598000

  Distance from Earth to sun, used in dawn/dusk calculations.

Functions

- double degToRad (double deg)

  Convert degrees to radians.
- int sign (double x)

  Checks sign of given value. If longitude value is negative, an offset is taken into account for dawn/duck calculations.
- double dawnCalc (int day, int month, int year, double latitude, double longitude)

  Calculation of approximate dawn time based on given day and location.
- double duskCalc (int day, int month, int year, double latitude, double longitude)

  Calculation of approximate dusk time based on given day and location.

5.3.1 Detailed Description

Desciprtion: Contains function declarations and constants used in approximate dawn and dusk times.

Author

: Shipra Vaidya

Date

last modified: 4/26/2020

5.3.2 Macro Definition Documentation

5.3.2.1 distSun

```
#define distSun 149598000
```

Distance from Earth to sun, used in dawn/dusk calculations.

5.3.2.2 pi

```
#define pi 3.1415926
```

Value of pi, used in dawn/dusk calculations.

5.3.2.3 planeDist

```
#define planeDist 23.45
```

Distance between x-y plane and ecliptic plane, used in dawn/dusk calculations.

5.3.2.4 radius

```
#define radius 6378
```

Size of radius of Earth, used in dawn/dusk calculation.

5.3.3 Function Documentation

5.3.3.1 dawnCalc()

```
double dawnCalc (
            int day,
            int month,
            int year,
            double latitude,
            double longitude )
```

Calculation of approximate dawn time based on given day and location.

Parameters

day	Numerical value, day of month.
month	Numerical value, month in year.
year	Numerical value, year.
latitude	Degree value of latitude from user's location.
longitude	Degree value of longitude from user's location.

Returns

double Returns approximate time of dawn on given date at given location.

5.3.3.2 degToRad()

```
double degToRad (
            double deg )
```

Convert degrees to radians.

Parameters

deg	Latitude or longitude value in degrees.

Returns

double Returns converted value in radians.

5.3.3.3 duskCalc()

```
double duskCalc (
            int day,
            int month,
            int year,
            double latitude,
            double longitude )
```

Calculation of approximate dusk time based on given day and location.

Parameters

day	Numerical value, day of month.
month	Numerical value, month in year.
year	Numerical value, year.
latitude	Degree value of latitude from user's location.
longitude	Degree value of longitude from user's location.

Returns

double Returns approximate time of dusk on given date at given location.

5.3.3.4 sign()

```
int sign (
            double x )
```

Checks sign of given value. If longitude value is negative, an offset is taken into account for dawn/duck calculations.

Parameters

x	Value of longitude.

Returns

int Returns 1 if positive, -1 if negative, to be multiplied in offset calculation.

5.4 include/decode_bluetooth.h File Reference

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
```

Macros

- #define DECODE_BLUETOOTH_H

Functions

- esp_err_t start_schedule_read (uint8_t *packet, uint16_t handle)

  Looks for "Begin Read" from the packet recieved. Saves channel number found in packet in saved_channel_num. Calls set_schedule read, passing handle parameter.
- esp_err_t set_schedule_read (uint16_t handle)

  Sets GATT characteristic with the respective handle to the value of a schedule. First sets the characteristic to the name of the first schedule. On the next read it will set the characteristic to the schedule information. This will repeat until there are no more schedules in the requested channel (saved_channel_num).
- esp_err_t decode_ble_schedule (uint8_t *packet)

  Converts byte packet in order to create schedule object. Recieves name from saved_name.
- esp_err_t decode_ble_schedule_name (uint8_t *packet, uint16_t length)

  Converts byte packet into string and saves value in saved_name.
- esp_err_t decode_ble_time (uint8_t *packet)

  Converts byte packet into UNIX time value and calls set_time function.
- esp_err_t decode_ble_direct (uint8_t *packet)

  Converts byte packet into channel, brightness, and color information. If RGB channel is written, calls set_color function. If normal LED channel is written, calls channel_on function.
- esp_err_t decode_ble_delete (uint8_t *packet)

  Converts byte packet into schedule name and channel. The schedule on the corresponding channel is then deleted using delete_schedule_by_name function.
- uint32_t get_Int32 (uint8_t *begin)

  Takes byte array and converts into a uint32 using the first 4 indices. Assumes Big-endian.

5.4.1 Detailed Description

Description:

Author

: Hunaid Puri

Date

last modified: 4/26/2020

5.4.2 Macro Definition Documentation

5.4.2.1 DECODE_BLUETOOTH_H

```
#define DECODE_BLUETOOTH_H
```

5.4.3 Function Documentation

5.4.3.1 decode_ble_delete()

```
esp_err_t decode_ble_delete (
            uint8_t * packet )
```

Converts byte packet into schedule name and channel. The schedule on the corresponding channel is then deleted using delete_schedule_by_name function.

Parameters

packet	recieved byte packet with schedule name to delete

Returns

esp_err_t

5.4.3.2 decode_ble_direct()

```
esp_err_t decode_ble_direct (
            uint8_t * packet )
```

Converts byte packet into channel, brightness, and color information. If RGB channel is written, calls set_color function. If normal LED channel is written, calls channel_on function.

Parameters

packet	recieved byte packet with led control information

Returns

esp_err_t

5.4.3.3 decode_ble_schedule()

```
esp_err_t decode_ble_schedule (
            uint8_t * packet )
```

Converts byte packet in order to create schedule object. Recieves name from saved_name.

Parameters

packet	recieved byte packet with schedule information

Returns

esp_err_t

5.4.3.4 decode_ble_schedule_name()

```
esp_err_t decode_ble_schedule_name (
            uint8_t * packet,
            uint16_t length )
```

Converts byte packet into string and saves value in saved_name.

Parameters

packet	recieved byte packet with scheudle name
length	length of recieved packet

Returns

esp_err_t

5.4.3.5 decode_ble_time()

```
esp_err_t decode_ble_time (
            uint8_t * packet )
```

Converts byte packet into UNIX time value and calls set_time function.

Parameters

packet	recieved byte packet with time information

Returns

esp_err_t

5.4.3.6 get_Int32()

```
uint32_t get_Int32 (
            uint8_t * begin )
```

Takes byte array and converts into a uint32 using the first 4 indices. Assumes Big-endian.

Parameters

begin	byte array

Returns

uint32_t

5.4.3.7 set_schedule_read()

```
esp_err_t set_schedule_read (
            uint16_t handle )
```

Sets GATT characteristic with the respective handle to the value of a schedule. First sets the characteristic to the name of the first schedule. On the next read it will set the characteristic to the schedule information. This will repeat until there are no more schedules in the requested channel (saved_channel_num).

Parameters

handle	handle for the characteristic that will be read

Returns

esp_err_t

5.4.3.8 start_schedule_read()

```
esp_err_t start_schedule_read (
            uint8_t * packet,
            uint16_t handle )
```

Looks for "Begin Read" from the packet recieved. Saves channel number found in packet in saved_channel_num. Calls set_schedule read, passing handle parameter.

Parameters

packet	recieved byte packet
handle	handle for the characteristic that will be read

Returns

esp_err_t

5.5 include/espsntp.h File Reference

```
#include <time.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_log.h"
#include "lwip/apps/sntp.h"
```

Functions

- void obtain_time (void)
- void set_time (uint32_t time)

5.5.1 Detailed Description

Description: Contains function declarations to start the process of obtaining time from an SNTP server. NOTE: not tested with latest version of project. Was initially developed very early in project and abandoned for other priorites.

Author

: Jesse Cannon (based on https://github.com/espressif/esp-idf/blob/5aa21584cfb4cfe4f2c7f625
_example_main.c)

Date

last modified: 4/26/2020

5.5.2 Function Documentation

5.5.2.1 obtain_time()

```
void obtain_time (
            void  )
```

5.5.2.2 set_time()

```
void set_time (
            uint32_t time )
```

5.6 include/gatts_table_creat_demo.h File Reference

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

Enumerations

- enum {
  IDX_SVC, IDX_CHAR_A, IDX_CHAR_VAL_A, IDX_CHAR_CFG_A,
  IDX_CHAR_B, IDX_CHAR_VAL_B, IDX_CHAR_C, IDX_CHAR_VAL_C,
  HRS_IDX_NB }

5.6.1 Enumeration Type Documentation

5.6.1.1 anonymous enum

```
anonymous enum
```

Enumerator

IDX_SVC	
IDX_CHAR_A	
IDX_CHAR_VAL_A	
IDX_CHAR_CFG_A	
IDX_CHAR_B	
IDX_CHAR_VAL_B	
IDX_CHAR_C	

5.7 include/http.h File Reference

```
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event_loop.h"
#include "esp_log.h"
#include "esp_http_server.h"
```

Functions

- esp_err_t init_http (httpd_handle_t server)

  Initializes a http server with other handlers defined the file.
- esp_err_t homepage_handler (httpd_req_t ∗req)

  Handler to retrieve home/direct control page. Accessed when server receives "/" or "/index.html".
- esp_err_t styles_handler (httpd_req_t ∗req)

  Handler to retrieve styles CSS page. Accessed when server receives "/styles.css".
- esp_err_t schedules_handler (httpd_req_t ∗req)

  Handler to retrieve schedules page. Accessed when server receives "/schedules.html".
- esp_err_t scripts_handler (httpd_req_t ∗req)

  Handler to retrieve scripts page. Accessed when server receives "/schedules.js".
- esp_err_t schedule_post_handler (httpd_req_t ∗req)

  Handler to create new schedule. Accessed when server receives or "/post_sch".
- esp_err_t favicon_ico_get_handler (httpd_req_t ∗req)

  Handler to retrieve favicon. Accessed when server receives "/favicon.ico" - DOES NOT WORK.
- esp_err_t time_post_handler (httpd_req_t ∗req)

  Handler to post time to esp32. Accessed when server receives "/time".
- esp_err_t direct_control_post_handler (httpd_req_t ∗req)

  Handler for direct control. Access when server receives "/direct_control".
- esp_err_t sch_data_post_handler (httpd_req_t ∗req)

  Handler to retrieve schedule data Access when server receives "/sch_data".
- void schTokenProcess (char ∗str)

  Function to process ";";"-delimited string. Parses out scheduing data and for schedule post handler.

5.7.1 Detailed Description

Description:

Author

: Andy Yeung

Date

: 4/26/2020

5.7.2 Function Documentation

5.7.2.1 direct_control_post_handler()

```
esp_err_t direct_control_post_handler (
            httpd_req_t * req )
```

Handler for direct control. Access when server receives "/direct_control".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.7.2.2 favicon_ico_get_handler()

```
esp_err_t favicon_ico_get_handler (
            httpd_req_t * req )
```

Handler to retrieve favicon. Accessed when server receives "/favicon.ico" - DOES NOT WORK.

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.7.2.3 homepage_handler()

```
esp_err_t homepage_handler (
            httpd_req_t * req )
```

Handler to retrieve home/direct control page. Accessed when server receives "/" or "/index.html".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.7.2.4 init_http()

```
esp_err_t init_http (
            httpd_handle_t server )
```

Initializes a http server with other handlers defined the file.

Parameters

server	A handle for the server.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.7.2.5 sch_data_post_handler()

```
esp_err_t sch_data_post_handler (
            httpd_req_t * req )
```

Handler to retrieve schedule data Access when server receives "/sch_data".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.7.2.6 schedule_post_handler()

```
esp_err_t schedule_post_handler (
            httpd_req_t * req )
```

Handler to create new schedule. Accessed when server receives or "/post_sch".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.7.2.7 schedules_handler()

```
esp_err_t schedules_handler (
            httpd_req_t * req )
```

Handler to retrieve schedules page. Accessed when server receives "/schedules.html".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.7.2.8 schTokenProcess()

```
void schTokenProcess (
            char * str )
```

Function to process ";"-delimited string. Parses out scheduing data and for schedule post handler.

Parameters

str	Received string

5.7.2.9 scripts_handler()

```
esp_err_t scripts_handler (
            httpd_req_t * req )
```

Handler to retrieve scripts page. Accessed when server receives "/schedules.js".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.7.2.10 styles_handler()

```
esp_err_t styles_handler (
            httpd_req_t * req )
```

Handler to retrieve styles CSS page. Accessed when server receives "/styles.css".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.7.2.11 time_post_handler()

```
esp_err_t time_post_handler (
            httpd_req_t * req )
```

Handler to post time to esp32. Accessed when server receives "/time".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.8 include/led.h File Reference

```
#include <stdio.h>
#include "esp_system.h"
#include "esp_log.h"
#include "driver/ledc.h"
```

Macros

- #define GPIO_CHANNEL_0 32

  Pin definitions for tech demo, needs to be changed for PCB.
- #define GPIO_CHANNEL_1 33
- #define GPIO_CHANNEL_2 25
- #define GPIO_CHANNEL_3 19
- #define GPIO_CHANNEL_4 18
- #define GPIO_CHANNEL_5 5
- #define NUM_CHANNELS 6

  Six pwm channels supported as either six individual channels, two RGB channels, or combination of two.

Functions

- esp_err_t shutdown_outputs (void)

  Shuts down all channels and prevents them from turning on again. Enter a permanent shutdown state until clear_↩ shutdown is called or device is reset.
- esp_err_t clear_shutdown (void)

  Clears a shutdown state, allowing outputs to be controlled again.
- void init_channels (void)

  Initialize channels to set all timer and channel configurations.
- void channel_on (uint8_t index, uint8_t brightness)

  Sets brightness of individual channel to turn that output on.
- void channel_off (uint8_t index)

  Sets brightness of individual channel to 0 to turn off channel output.
- void set_color (uint8_t index, uint16_t r, uint16_t g, uint16_t b, uint8_t brightness)

  Controls RGB channel by changing R, G, and B based off chosen brightness.

5.8.1 Detailed Description

Description: Contains function declarations and settings to configure GPIO outputs to be used as PWM control for LED strips.

Author

: Primary: Shipra Vaidya, Secondary: Jesse Cannon (shutdown features)

Date

last modified: 4/26/2020

5.8.2 Macro Definition Documentation

5.8.2.1 GPIO_CHANNEL_0

```
#define GPIO_CHANNEL_0 32
```

Pin definitions for tech demo, needs to be changed for PCB.

5.8.2.2 GPIO_CHANNEL_1

```
#define GPIO_CHANNEL_1 33
```

5.8.2.3 GPIO_CHANNEL_2

```
#define GPIO_CHANNEL_2 25
```

5.8.2.4 GPIO_CHANNEL_3

```
#define GPIO_CHANNEL_3 19
```

5.8.2.5 GPIO_CHANNEL_4

```
#define GPIO_CHANNEL_4 18
```

5.8.2.6 GPIO_CHANNEL_5

```
#define GPIO_CHANNEL_5 5
```

5.8.2.7 NUM_CHANNELS

```
#define NUM_CHANNELS 6
```

Six pwm channels supported as either six individual channels, two RGB channels, or combination of two.

5.8.3 Function Documentation

5.8.3.1 channel_off()

```
void channel_off (
            uint8_t index )
```

Sets brightness of individual channel to 0 to turn off channel output.

Parameters

index	Value between 0 and 5 corresponding to channel being controlled.

5.8.3.2 channel_on()

```
void channel_on (
            uint8_t index,
            uint8_t brightness )
```

Sets brightness of individual channel to turn that output on.

Parameters

index	Value between 0 and 5 corresponding to channel being controlled.
brightness	Value of brightness chosen per channel.

5.8.3.3 clear_shutdown()

```
esp_err_t clear_shutdown (
            void  )
```

Clears a shutdown state, allowing outputs to be controlled again.

Returns

esp_err_t ESP_OK on success, ESP_FAIL if not in shutdown state.

5.8.3.4 init_channels()

```
void init_channels (
            void  )
```

Initialize channels to set all timer and channel configurations.

5.8.3.5 set_color()

```
void set_color (
            uint8_t index,
            uint16_t r,
            uint16_t g,
            uint16_t b,
            uint8_t brightness )
```

Controls RGB channel by changing R, G, and B based off chosen brightness.

Parameters

index	Value between 0 and 1 correnspondingto two supported RGB channel outputs repectively.
r	R value of chosen color.
g	G value of chosen color.
b	B value of chosen color.
brightness	Value of brightness chosen for RGB channel.

5.8.3.6 shutdown_outputs()

```
esp_err_t shutdown_outputs (
            void  )
```

Shuts down all channels and prevents them from turning on again. Enter a permanent shutdown state until clear↩ _shutdown is called or device is reset.

Returns

esp_err_t ESP_OK on success, ESP_FAIL if already in shutdown state.

5.9 include/main.h File Reference

5.9.1 Detailed Description

Description: Contains code version of latest build. Used by memory subsystem to keep old versions separated from new versions so that no conflicts occur with updates.

Author

: Jesse Cannon

Date

last modified: 4/26/2020

5.10 include/measurement.h File Reference

```
#include "driver/gpio.h"
#include "driver/adc.h"
#include "driver/dac.h"
#include "esp_system.h"
#include "esp_log.h"
#include "pin_defs.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "led.h"
#include "memory.h"
```

Functions

- esp_err_t init_oc (void)

  Inits the overcurrent and overvoltage detection systems. The previously used values will be set as the level, with defaults of 24 Volts and 2 Amps triggering faults.
- esp_err_t set_current_level (double ampLimit)

  Set the current level that will trigger a fault.
- esp_err_t set_voltage_level (double voltLimit)

  Set the voltage level that will trigger a fault.
- uint8_t isCurrentFault ()

  Determines if the device is in shutdown from an overcurrent condition.
- uint8_t isVoltageFault ()

  Determines if the device is in shutdown from an overvoltage condition.
- esp_err_t clearFaults ()

  Clears all faults from the system and allows outputs to be controlled again.

5.10.1 Detailed Description

Description: Contains function declarations for OC/OV subsystem. Intended to set levels and take actions for overcurrent and overvoltage conditions. NOTE: this system is basically just an outline. Development was not completed b/c of a lack of hardware and time constraints.

Author

: Jesse Cannon

Date

last modified: 4/26/2020

5.10.2 Function Documentation

5.10.2.1 clearFaults()

```
esp_err_t clearFaults ( )
```

Clears all faults from the system and allows outputs to be controlled again.

Returns

esp_err_t ESP_OK if successfully re-enables outputs. ESP_FAIL if error occurs.

5.10.2.2 init_oc()

```
esp_err_t init_oc (
            void  )
```

Inits the overcurrent and overvoltage detection systems. The previously used values will be set as the level, with defaults of 24 Volts and 2 Amps triggering faults.

Returns

esp_err_t

5.10.2.3 isCurrentFault()

```
uint8_t isCurrentFault ( )
```

Determines if the device is in shutdown from an overcurrent condition.

Returns

uint8_t A value of 1 means that there is currently a fault. A value of 0 means there is no fault

5.10.2.4 isVoltageFault()

```
uint8_t isVoltageFault ( )
```

Determines if the device is in shutdown from an overvoltage condition.

Returns

uint8_t A value of 1 means that there is currently a fault. A value of 0 means there is no fault

5.10.2.5 set_current_level()

```
esp_err_t set_current_level (
            double ampLimit )
```

Set the current level that will trigger a fault.

Parameters

ampLimit	The current level to be used to determine faults, such as 1.1 Amps.

Returns

esp_err_t ESP_OK if the new level was successfully set, else if error occurred.

5.10.2.6 set_voltage_level()

```
esp_err_t set_voltage_level (
            double voltLimit )
```

Set the voltage level that will trigger a fault.

Parameters

voltLimit	The voltage level to be used to determine faults, such as 12.5 Volts.

Returns

esp_err_t ESP_OK if the new level was successfully set, else if error occurred.

5.11 include/memory.h File Reference

```
#include "ArduinoJson-v6.14.1.h"
#include <stdio.h>
#include <string.h>
#include <sys/unistd.h>
#include <sys/stat.h>
#include <dirent.h>
#include "main.h"
#include "esp_err.h"
#include "esp_log.h"
#include "esp_spiffs.h"
#include "scheduler.h"
```

Macros

- #define SETTINGS_BUFFER_SIZE 512

  The size of the buffer used to store settings. Increase this if storing a lot of settings.

Functions

- esp_err_t init_memory (void)

  Starts the memory system. Call this before calling any other memory related functions.
- esp_err_t store_schedules (void)

  Stores all currently running schedules in persistent memory.
- esp_err_t recall_schedules (void)

Recall schedules from persistent memory. All schedules recalled will now be located in the tespective channel's schedule list.

- esp_err_t clear_schedule_data (void)

  Clears all schedules from persistent memory. Does not delete the schedules from the list.

- esp_err_t store_setting_string (const char ∗name, char ∗setting)

  Persistently store a string setting.

- esp_err_t store_setting_int (const char ∗name, int setting)

  Persistently store an integer setting.

- esp_err_t store_setting_byte (const char ∗name, uint8_t setting)

  Persistently store a byte setting.

- esp_err_t store_setting_double (const char ∗name, double setting)

  Persistently store a double setting.

- esp_err_t get_setting_string (const char ∗name, char ∗setting)

  Recall a persistent string setting.

- esp_err_t get_setting_int (const char ∗name, int ∗setting)

  Recall a persistent integer setting.

- esp_err_t get_setting_byte (const char ∗name, uint8_t ∗setting)

  Recall a byte string setting. Uint8_t is equivalent to unsigned char.

- esp_err_t get_setting_double (const char ∗name, double ∗setting)

  Recall a persistent double setting. Exact storage depends on ArduinoJson configuration, but typically a maximum of 9 decimal places with rounding. No trailing zeros are included.

- esp_err_t clear_setting_data (void)

  Clear settings from persistent memory and RAM.

5.11.1 Detailed Description

Description: Contains function declarations and settings used by the memory subsystem. Functions are primarily used by other files to get and retrieve values from persistent memory.

Author

: Jesse Cannon

Date

last modified: 4/26/2020

5.11.2 Macro Definition Documentation

5.11.2.1 SETTINGS_BUFFER_SIZE

```
#define SETTINGS_BUFFER_SIZE 512
```

The size of the buffer used to store settings. Increase this if storing a lot of settings.

5.11.3 Function Documentation

5.11.3.1 clear_schedule_data()

```
esp_err_t clear_schedule_data (
            void  )
```

Clears all schedules from persistent memory. Does not delete the schedules from the list.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.11.3.2 clear_setting_data()

```
esp_err_t clear_setting_data (
            void  )
```

Clear settings from persistent memory and RAM.

Returns

esp_err_t ESP_OK on success. ESP_FAIL if error occurrs.

5.11.3.3 get_setting_byte()

```
esp_err_t get_setting_byte (
            const char * name,
            uint8_t * setting )
```

Recall a byte string setting. Uint8_t is equivalent to unsigned char.

Parameters

name	The name of the setting to recall.
setting	The output of that setting. Set to NULL if not found, the value if found.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail if not found or an error occurred.

5.11.3.4 get_setting_double()

```
esp_err_t get_setting_double (
            const char * name,
            double * setting )
```

Recall a persistent double setting. Exact storage depends on ArduinoJson configuration, but typically a maximum of 9 decimal places with rounding. No trailing zeros are included.

Parameters

name	The name of the setting to recall.
setting	The output of that setting. Set to NULL if not found, the value if found.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail if not found or an error occurred.

5.11.3.5 get_setting_int()

```
esp_err_t get_setting_int (
            const char * name,
            int * setting )
```

Recall a persistent integer setting.

Parameters

name	The name of the setting to recall.
setting	The output of that setting. Set to NULL if not found, the value if found.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail if not found or an error occurred.

5.11.3.6 get_setting_string()

```
esp_err_t get_setting_string (
            const char * name,
            char * setting )
```

Recall a persistent string setting.

Parameters

name	The name of the setting to recall.
setting	The output of that setting. Set to NULL if not found, the value if found.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail if not found or an error occurred.

5.11.3.7 init_memory()

```
esp_err_t init_memory (
            void  )
```

Starts the memory system. Call this before calling any other memory related functions.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.11.3.8 recall_schedules()

```
esp_err_t recall_schedules (
            void  )
```

Recall schedules from persistent memory. All schedules recalled will now be located in the tespective channel's schedule list.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.11.3.9 store_schedules()

```
esp_err_t store_schedules (
            void  )
```

Stores all currently running schedules in persistent memory.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.11.3.10 store_setting_byte()

```
esp_err_t store_setting_byte (
            const char * name,
            uint8_t setting )
```

Persistently store a byte setting.

Parameters

name	The name of the setting to store. If the setting with that name already exists it is overwritten with the new value.
setting	The setting to associate with the given name. Uint8_t is equivalent to an unsigned char.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.11.3.11 store_setting_double()

```
esp_err_t store_setting_double (
            const char * name,
            double setting )
```

Persistently store a double setting.

Parameters

name	The name of the setting to store. If the setting with that name already exists it is overwritten with the new value.
setting	The setting to associate with the given name. Exact storage depends on ArduinoJson configuration, but typically a maximum of 9 decimal places with rounding. No trailing zeros are included

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.11.3.12 store_setting_int()

```
esp_err_t store_setting_int (
            const char * name,
            int setting )
```

Persistently store an integer setting.

Parameters

name	The name of the setting to store. If the setting with that name already exists it is overwritten with the new value.
setting	The setting to associate with the given name.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.11.3.13 store_setting_string()

```
esp_err_t store_setting_string (
            const char * name,
            char * setting )
```

Persistently store a string setting.

Parameters

name	The name of the setting to store. If the setting with that name already exists it is overwritten with the new value.
setting	The setting to associate with the given name.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.12 include/pin_defs.h File Reference

Macros

- #define CH3_HIGH 39
- #define CH3_LOW 36
- #define DAC1 1
- #define DAC2 26
- #define OC_ENABLE GPIO_NUM_22
- #define OC_ALERT GPIO_NUM_26
- #define OC_LATCH
- #define OC_LIMIT DAC1

5.12.1 Detailed Description

Description: Pin definitions for external hardware. Used by measurement subsystem for INA300 for overcurrent detection

Author

: Jesse Cannon

Date

last modified: 4/26/2020

5.12.2 Macro Definition Documentation

5.12.2.1 CH3_HIGH

```
#define CH3_HIGH 39
```

5.12.2.2 CH3_LOW

```
#define CH3_LOW 36
```

5.12.2.3 DAC1

```
#define DAC1 1
```

5.12.2.4 DAC2

```
#define DAC2 26
```

5.12.2.5 OC_ALERT

```
#define OC_ALERT GPIO_NUM_26
```

5.12.2.6 OC_ENABLE

```
#define OC_ENABLE GPIO_NUM_22
```

5.12.2.7 OC_LATCH

```
#define OC_LATCH
```

5.12.2.8 OC_LIMIT

```
#define OC_LIMIT DAC1
```

5.13 include/rtcdefine.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_timer.h"
#include "esp_log.h"
#include <time.h>
#include <sys/time.h>
#include "driver/spi_master.h"
#include "esp_spi_flash.h"
#include "sdkconfig.h"
```

Macros

- #define HSPI_MISO 12
- #define HSPI_MOSI 13
- #define HSPI_CLK 14
- #define HSPI_CS 15
- #define EEREAD 0x03
- #define EEWRITE 0x02
- #define EEWRDI 0x04
- #define EEWREN 0x06
- #define SRREAD 0x05
- #define SSWRITE 0x01
- #define RTC_READ 0x13
- #define RTC_WRITE 0x12
- #define RTC_UNLOCK 0x14
- #define IDWRITE 0x32
- #define IDREAD 0x33
- #define CLRRAM 0x54
- #define RTCHSEC 0x00
- #define RTCSEC 0x01
- #define RTCMIN 0x02
- #define RTCHOUR 0x03
- #define RTCWKDAY 0x04
- #define RTCDATE 0x05
- #define RTCMTH 0x06
- #define RTCYEAR 0x07
- #define CONTROL 0x08
- #define OSCTRIM 0x09
- #define PWRDNMIN 0x18
- #define PWRDNHOUR 0x19
- #define PWRDNDATE 0x1A

- #define PWRDNMONTH 0x1B
- #define ST 0x80
- #define LPYR 0x20
- #define PM 0x20
- #define H12 0x40
- #define TRIMSIGN 0x80
- #define OUT 0x80
- #define SQWEN 0x40
- #define OSCRUN 0x20
- #define VBATEN 0x08
- #define PWRFAIL 0x10
- #define RTC_TAG "RTC"

Functions

- esp_err_t setTime (const struct tm *time)

  Set the time of the external RTC to the specified value.
- esp_err_t getTime (struct tm *outTime)

  Get the current time on the external RTC.
- void RTCHandler (void *pvParms)

  The RTOS task that handles RTC related actions. This task syncrhonizes the esp32's and RTC's time.
- esp_err_t ST_StartRTCHandler (void)

  Starts the RTOS task that handles RTC and esp32 time synchronization.

Variables

- TaskHandle_t RTC_Handle

  The task handle for the RTC task. If this field is != NULL then task is currently running. Can be used to stop task with RTOS functions.
- struct tm * currTime

  A copy of the most recent time obtained from the RTC. Is not guaranteed to be accurate to the actual time stored on the RTC. This value is updated as often as the RTC task runs.

5.13.1 Detailed Description

Description: Contains function and pin definitions for the time manager subsystem. Controls external RTC MCP79510. see: http://ww1.microchip.com/downloads/en/DeviceDoc/MCP7951X-MC↩P7952X-Battery-Backed-SPI-RTCC-20002300C.pdf

Author

: Lead: Shipra Vaidya, Seoncdary: Jesse Cannon

Date

last modified: 4/26/2020

5.13.2 Macro Definition Documentation

5.13.2.1 CLRRAM

```
#define CLRRAM 0x54
```

5.13.2.2 CONTROL

```
#define CONTROL 0x08
```

5.13.2.3 EEREAD

```
#define EEREAD 0x03
```

5.13.2.4 EEWRDI

```
#define EEWRDI 0x04
```

5.13.2.5 EEWREN

```
#define EEWREN 0x06
```

5.13.2.6 EEWRITE

```
#define EEWRITE 0x02
```

5.13.2.7 H12

```
#define H12 0x40
```

5.13.2.8 HSPI_CLK

```
#define HSPI_CLK 14
```

5.13.2.9 HSPI_CS

```
#define HSPI_CS 15
```

5.13.2.10 HSPI_MISO

```
#define HSPI_MISO 12
```

5.13.2.11 HSPI_MOSI

```
#define HSPI_MOSI 13
```

5.13.2.12 IDREAD

```
#define IDREAD 0x33
```

5.13.2.13 IDWRITE

```
#define IDWRITE 0x32
```

5.13.2.14 LPYR

```
#define LPYR 0x20
```

5.13.2.15 OSCRUN

```
#define OSCRUN 0x20
```

5.13.2.16 OSCTRIM

```
#define OSCTRIM 0x09
```

5.13.2.17 OUT

```
#define OUT 0x80
```

5.13.2.18 PM

```
#define PM 0x20
```

5.13.2.19 PWRDNDATE

```
#define PWRDNDATE 0x1A
```

5.13.2.20 PWRDNHOUR

```
#define PWRDNHOUR 0x19
```

5.13.2.21 PWRDNMIN

```
#define PWRDNMIN 0x18
```

5.13.2.22 PWRDNMONTH

```
#define PWRDNMONTH 0x1B
```

5.13.2.23 PWRFAIL

```
#define PWRFAIL 0x10
```

5.13.2.24 RTC_READ

```
#define RTC_READ 0x13
```

5.13.2.25 RTC_TAG

```
#define RTC_TAG "RTC"
```

5.13.2.26 RTC_UNLOCK

```
#define RTC_UNLOCK 0x14
```

5.13.2.27 RTC_WRITE

```
#define RTC_WRITE 0x12
```

5.13.2.28 RTCDATE

```
#define RTCDATE 0x05
```

5.13.2.29 RTCHOUR

```
#define RTCHOUR 0x03
```

5.13.2.30 RTCHSEC

```
#define RTCHSEC 0x00
```

5.13.2.31 RTCMIN

```
#define RTCMIN 0x02
```

5.13.2.32 RTCMTH

```
#define RTCMTH 0x06
```

5.13.2.33 RTCSEC

```
#define RTCSEC 0x01
```

5.13.2.34 RTCWKDAY

```
#define RTCWKDAY 0x04
```

5.13.2.35 RTCYEAR

```
#define RTCYEAR 0x07
```

5.13.2.36 SQWEN

```
#define SQWEN 0x40
```

5.13.2.37 SRREAD

```
#define SRREAD 0x05
```

5.13.2.38 SSWRITE

```
#define SSWRITE 0x01
```

5.13.2.39 ST

```
#define ST 0x80
```

5.13.2.40 TRIMSIGN

```
#define TRIMSIGN 0x80
```

5.13.2.41 VBATEN

```
#define VBATEN 0x08
```

5.13.3 Function Documentation

5.13.3.1 getTime()

```
esp_err_t getTime (
            struct tm * outTime )
```

Get the current time on the external RTC.

Parameters

outTime	The current time on the RTC will be stored in this value. Check the return type against ESP_OK to ensure this value was set correctly.

Returns

esp_err_t ESP_OK on success, else on failure.

5.13.3.2 RTCHandler()

```
void RTCHandler (
            void * pvParms )
```

The RTOS task that handles RTC related actions. This task syncrhonizes the esp32's and RTC's time.

Parameters

pvParms	Required parameter of RTOS tasks. Not used in this task.

5.13.3.3 setTime()

```
esp_err_t setTime (
            const struct tm * time )
```

Set the time of the external RTC to the specified value.

Parameters

time	The time to be set on the device.

Returns

esp_err_t ESP_OK on success, else on failure.

5.13.3.4 ST_StartRTCHandler()

```
esp_err_t ST_StartRTCHandler (
            void  )
```

Starts the RTOS task that handles RTC and esp32 time synchronization.

Returns

esp_err_t ESP_OK if task started successfully, else on failure.

5.13.4 Variable Documentation

5.13.4.1 currTime

```
struct tm* currTime
```

A copy of the most recent time obtained from the RTC. Is not guaranteed to be accurate to the actual time stored on the RTC. This value is updated as often as the RTC task runs.

5.13.4.2 RTC_Handle

```
TaskHandle_t RTC_Handle
```

The task handle for the RTC task. If this field is != NULL then task is currently running. Can be used to stop task with RTOS functions.

5.14 include/schedule_object.h File Reference

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
```

Classes

- struct Schedule_Object

  data object that specified parameters a schedule can contain
- struct Node

  Nodes of a singly linked list data structure.

Typedefs

- typedef struct Schedule_Object schedule_object

  data object that specified parameters a schedule can contain
- typedef struct Node List

  Nodes of a singly linked list data structure.

5.14.1 Detailed Description

Description: Contains the definition of the data object used for schedules in the scheduler subsystem. Defines a simple linked list data structure to contain schedule data object.

Author

: Jesse Cannon

Date

last modified: 4/26/2020

5.14.2 Typedef Documentation

5.14.2.1 List

```
typedef struct Node List
```

Nodes of a singly linked list data structure.

5.14.2.2 schedule_object

```
typedef struct Schedule_Object schedule_object
```

data object that specified parameters a schedule can contain

5.15 include/scheduler.h File Reference

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_log.h"
#include <time.h>
#include <string.h>
#include "schedule_object.h"
#include "led.h"
#include "dawndusk.h"
#include "memory.h"
```

Functions

- void init_schedule (void)

  Init the scheduler code. Starts the background task if not already started. If this is not called the schedules will not run.
- esp_err_t create_schedule (uint8_t channel, schedule_object s)

  Create a schedule to associate with the specified channel. If the schedule already exists then it is replaced with the new schedule.
- esp_err_t delete_schedule_by_id (uint8_t channel, uint8_t ID)

  Delete a schedule by ID.
- esp_err_t delete_schedule_by_name (uint8_t channel, char ∗name)

  Delete a schedule by name.
- esp_err_t disable_schedule_by_id (uint8_t channel, uint8_t ID)

  Disable a schedule by ID. The schedule remains in the linked list but does not run or perform updates.
- esp_err_t disable_schedule_by_name (uint8_t channel, char ∗name)

  Disable a schedule by name. The schedule remains in the linked list but does not run or perform updates.
- esp_err_t enable_schedule_by_id (uint8_t channel, uint8_t ID)

  Enable a schedule by ID. If the schedule was disabled, the schedule now runs and performs updates.
- esp_err_t enable_schedule_by_name (uint8_t channel, char ∗name)

  Enable a schedule by name. If the schedule was disabled, the schedule now runs and performs updates.
- esp_err_t disable_all_schedules (void)

  Disables all schedules on all channels. The schedules remain located in memory but no longer run or perform updates.
- esp_err_t enable_all_schedules (void)

  Enable all schedules on all channels. If the schedule was disabled, the schedule now runs and performs updates.
- esp_err_t delete_all_schedules (void)

  Delete all schedules on all channels.
- esp_err_t get_schedule_names (uint8_t channel, char ∗&out)

  Get a json string of all schedules and their enabled status for a given channel.
- esp_err_t get_schedule (uint8_t channel, char ∗name, schedule_object ∗out)

  Get the schedule object specified.

Variables

- List ∗ schedules [NUM_CHANNELS]

  The linked list associated with each channel. Can be used directly instead of scheduler functions if so desired.

5.15.1 Detailed Description

Description: Contains functions available to start and manipulate the scheduler subsystem.

Author

: Jesse Cannon

Date

: 4/26/2020

5.15.2 Function Documentation

5.15.2.1 create_schedule()

```
esp_err_t create_schedule (
            uint8_t channel,
            schedule_object s )
```

Create a schedule to associate with the specified channel. If the schedule already exists then it is replaced with the new schedule.

Parameters

channel	The channel that the schedule will be placed on.
s	The schedule object that will be placed.

Returns

esp_err_t Returns ESP_OK on success. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ER↩ R_NO_MEM if unable to allocate schedule.

5.15.2.2 delete_all_schedules()

```
esp_err_t delete_all_schedules (
            void  )
```

Delete all schedules on all channels.

Returns

esp_err_t ESP_OK on successful deletion.

5.15.2.3 delete_schedule_by_id()

```
esp_err_t delete_schedule_by_id (
            uint8_t channel,
            uint8_t ID )
```

Delete a schedule by ID.

Parameters

channel	The channel the schedule is located on.
ID	The ID of the schedule to delete.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.15.2.4 delete_schedule_by_name()

```
esp_err_t delete_schedule_by_name (
            uint8_t channel,
            char * name )
```

Delete a schedule by name.

Parameters

channel	The channel the schedule is located on.
name	The name of the schedule to delete.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.15.2.5 disable_all_schedules()

```
esp_err_t disable_all_schedules (
            void  )
```

Disables all schedules on all channels. The schedules remain located in memory but no longer run or perform updates.

Returns

esp_err_t ESP_OK on successful disable.

5.15.2.6 disable_schedule_by_id()

```
esp_err_t disable_schedule_by_id (
            uint8_t channel,
            uint8_t ID )
```

Disable a schedule by ID. The schedule remains in the linked list but does not run or perform updates.

Parameters

channel	The channel the schedule is located on.
ID	The name of the schedule to disable.

Returns

esp_err_t ESP_OK on successful disable. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_E↩ RR_NOT_FOUND if schedule not found

5.15.2.7 disable_schedule_by_name()

```
esp_err_t disable_schedule_by_name (
            uint8_t channel,
            char * name )
```

Disable a schedule by name. The schedule remains in the linked list but does not run or perform updates.

Parameters

channel	The channel the schedule is located on.
name	The name of the schedule to disable.

Returns

esp_err_t ESP_OK on successful disable. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_E↩ RR_NOT_FOUND if schedule not found

5.15.2.8 enable_all_schedules()

```
esp_err_t enable_all_schedules (
            void  )
```

Enable all schedules on all channels. If the schedule was disabled, the schedule now runs and performs updates.

Returns

esp_err_t ESP_OK on successful disable.

5.15.2.9 enable_schedule_by_id()

```
esp_err_t enable_schedule_by_id (
            uint8_t channel,
            uint8_t ID )
```

Enable a schedule by ID. If the schedule was disabled, the schedule now runs and performs updates.

Parameters

channel	The channel the schedule is located on.
ID	The ID of the schedule to enable.

Returns

esp_err_t ESP_OK on successful enable. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_E↩ RR_NOT_FOUND if schedule not found

5.15.2.10 enable_schedule_by_name()

```
esp_err_t enable_schedule_by_name (
            uint8_t channel,
            char * name )
```

Enable a schedule by name. If the schedule was disabled, the schedule now runs and performs updates.

Parameters

channel	The channel the schedule is located on.
name	The name of the schedule to enable.

Returns

esp_err_t ESP_OK on successful enable. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_E↩ RR_NOT_FOUND if schedule not found

5.15.2.11 get_schedule()

```
esp_err_t get_schedule (
            uint8_t channel,
            char * name,
            schedule_object * out )
```

Get the schedule object specified.

Parameters

channel	The channel the schedule is located on.
name	The name of the schedule to find.
out	The schedule returned. NULL if failure occurred.

Returns

esp_err_t ESP_OK if successful. ESP_ERR_INVALID_ARG if channel is invalid. ESP_ERR_NOT_FOUND if the schedule could not be found.

5.15.2.12 get_schedule_names()

```
esp_err_t get_schedule_names (
            uint8_t channel,
            char *& out )
```

Get a json string of all schedules and their enabled status for a given channel.

Parameters

channel	The channel to get the schedule names and status from.
out	The json string that is returned. Format is '{ "Name1":0, "Name2":1, "Name3":0 }'

Returns

esp_err_t ESP_OK if successful. ESP_ERR_INVALID_ARG if channel is invalid.

5.15.2.13 init_schedule()

```
void init_schedule (
            void  )
```

Init the scheduler code. Starts the background task if not already started. If this is not called the schedules will not run.

5.15.3 Variable Documentation

5.15.3.1 schedules

```
List* schedules[NUM_CHANNELS]
```

The linked list associated with each channel. Can be used directly instead of scheduler functions if so desired.

5.16 include/wifi.h File Reference

```
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event_loop.h"
#include "esp_log.h"
```

Macros

- #define MIN(x, y) ((x)<(y)?(x):(y))

Functions

- void wifi_init_softap ()
- void wifi_init_sta ()

5.16.1 Detailed Description

Description: Contains function declarations to connect to a WiFi networking using SSID defined in wifi.cpp

Author

: Lead: Andy Yeung, Secondary: Jesse Cannon

Date

last modified: 4/26/2020

5.16.2 Macro Definition Documentation

5.16.2.1 MIN

```
#define MIN(
              x,
              y ) ((x)<(y)?(x):(y))
```

5.16.3 Function Documentation

5.16.3.1 wifi_init_softap()

```
void wifi_init_softap ( )
```

5.16.3.2 wifi_init_sta()

```
void wifi_init_sta ( )
```

5.17 README.md File Reference

5.18 src/bleSL.cpp File Reference

GATT server demo code used as base from esp-idf. Most modifications made in Init_Bluetooth function and gatts↩ _profile_event_handler callback.

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "esp_bt.h"
#include "esp_gap_ble_api.h"
#include "esp_gatts_api.h"
#include "esp_bt_main.h"
#include "gatts_table_creat_demo.h"
#include "esp_gatt_common_api.h"
#include "bleSL.h"
#include "decode_bluetooth.h"
```

Classes

- struct prepare_type_env_t
- struct gatts_profile_inst

Macros

- #define GATTS_TABLE_TAG "GATTS_TABLE_DEMO"
- #define PROFILE_NUM 1
- #define PROFILE_APP_IDX 0
- #define ESP_APP_ID 0x55
- #define SAMPLE_DEVICE_NAME "ESP_GATTS_DEMO"
- #define SVC_INST_ID 0
- #define GATTS_DEMO_CHAR_VAL_LEN_MAX 500
- #define PREPARE_BUF_MAX_SIZE 1024
- #define CHAR_DECLARATION_SIZE (sizeof(uint8_t))
- #define ADV_CONFIG_FLAG (1 << 0)
- #define SCAN_RSP_CONFIG_FLAG (1 << 1)
- #define CONFIG_SET_RAW_ADV_DATA

Functions

- void example_prepare_write_event_env (esp_gatt_if_t gatts_if, prepare_type_env_t *prepare_write_env, esp_ble_gatts_cb_param_t *param)
- void example_exec_write_event_env (prepare_type_env_t *prepare_write_env, esp_ble_gatts_cb_param_t *param)
- void Init_Bluetooth (void)

  Initializes GATT Server.

Variables

- uint16_t heart_rate_handle_table [HRS_IDX_NB]

5.18.1 Detailed Description

GATT server demo code used as base from esp-idf. Most modifications made in Init_Bluetooth function and gatts↩ _profile_event_handler callback.

Author

Hunaid Puri (hunaid14@gmail.com)

Version

0.1

Date

2020-04-26

Copyright

Copyright (c) 2020

5.18.2 Macro Definition Documentation

5.18.2.1 ADV_CONFIG_FLAG

```
#define ADV_CONFIG_FLAG (1 << 0)
```

5.18.2.2 CHAR_DECLARATION_SIZE

```
#define CHAR_DECLARATION_SIZE (sizeof(uint8_t))
```

5.18.2.3 CONFIG_SET_RAW_ADV_DATA

```
#define CONFIG_SET_RAW_ADV_DATA
```

5.18.2.4 ESP_APP_ID

```
#define ESP_APP_ID 0x55
```

5.18.2.5 GATTS_DEMO_CHAR_VAL_LEN_MAX

```
#define GATTS_DEMO_CHAR_VAL_LEN_MAX 500
```

5.18.2.6 GATTS_TABLE_TAG

```
#define GATTS_TABLE_TAG "GATTS_TABLE_DEMO"
```

5.18.2.7 PREPARE_BUF_MAX_SIZE

```
#define PREPARE_BUF_MAX_SIZE 1024
```

5.18.2.8 PROFILE_APP_IDX

```
#define PROFILE_APP_IDX 0
```

5.18.2.9 PROFILE_NUM

```
#define PROFILE_NUM 1
```

5.18.2.10 SAMPLE_DEVICE_NAME

```
#define SAMPLE_DEVICE_NAME "ESP_GATTS_DEMO"
```

5.18.2.11 SCAN_RSP_CONFIG_FLAG

```
#define SCAN_RSP_CONFIG_FLAG (1 << 1)
```

5.18.2.12 SVC_INST_ID

```
#define SVC_INST_ID 0
```

5.18.3 Function Documentation

5.18.3.1 example_exec_write_event_env()

```
void example_exec_write_event_env (
            prepare_type_env_t * prepare_write_env,
            esp_ble_gatts_cb_param_t * param )
```

5.18.3.2 example_prepare_write_event_env()

```
void example_prepare_write_event_env (
            esp_gatt_if_t gatts_if,
            prepare_type_env_t * prepare_write_env,
            esp_ble_gatts_cb_param_t * param )
```

5.18.3.3 Init_Bluetooth()

```
void Init_Bluetooth (
            void  )
```

Initializes GATT Server.

5.18.4 Variable Documentation

5.18.4.1 heart_rate_handle_table

```
uint16_t heart_rate_handle_table[HRS_IDX_NB]
```

5.19 src/dawndusk.cpp File Reference

```
#include "dawndusk.h"
```

Functions

- double degToRad (double deg)

  Convert degrees to radians.
- int sign (double x)

  Checks sign of given value. If longitude value is negative, an offset is taken into account for dawn/duck calculations.
- double dawnCalc (int day, int month, int year, double latitude, double longitude)

  Calculation of approximate dawn time based on given day and location.
- double duskCalc (int day, int month, int year, double latitude, double longitude)

  Calculation of approximate dusk time based on given day and location.

5.19.1 Detailed Description

Description: Contains function definitions to approximate dawn and dusk times based on latitude and longitude. This article was first published on Quantitative Ecology, and adapted from R-bloggers.

Author

: Shipra Vaidya

Date

last modified: 4/26/2020

5.19.2 Function Documentation

5.19.2.1 dawnCalc()

```
double dawnCalc (
            int day,
            int month,
            int year,
            double latitude,
            double longitude )
```

Calculation of approximate dawn time based on given day and location.

Parameters

day	Numerical value, day of month.
month	Numerical value, month in year.
year	Numerical value, year.
latitude	Degree value of latitude from user's location.
longitude	Degree value of longitude from user's location.

Returns

double Returns approximate time of dawn on given date at given location.

5.19.2.2 degToRad()

```
double degToRad (
            double deg )
```

Convert degrees to radians.

Parameters

deg	Latitude or longitude value in degrees.

Returns

double Returns converted value in radians.

5.19.2.3 duskCalc()

```
double duskCalc (
            int day,
            int month,
            int year,
            double latitude,
            double longitude )
```

Calculation of approximate dusk time based on given day and location.

Parameters

day	Numerical value, day of month.
month	Numerical value, month in year.
year	Numerical value, year.
latitude	Degree value of latitude from user's location.
longitude	Degree value of longitude from user's location.

Returns

double Returns approximate time of dusk on given date at given location.

5.19.2.4 sign()

```
int sign (
            double x )
```

Checks sign of given value. If longitude value is negative, an offset is taken into account for dawn/duck calculations.

Parameters

x	Value of longitude.

Returns

int Returns 1 if positive, -1 if negative, to be multiplied in offset calculation.

5.20 src/decode_bluetooth.cpp File Reference

This file contains functions used to decode/set byte packets from GATT server.

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "decode_bluetooth.h"
#include "scheduler.h"
#include "schedule_object.h"
#include "espsntp.h"
#include "esp_gatts_api.h"
```

Functions

- esp_err_t start_schedule_read (uint8_t *packet, uint16_t handle)

  Looks for "Begin Read" from the packet recieved. Saves channel number found in packet in saved_channel_num. Calls set_schedule read, passing handle parameter.
- esp_err_t set_schedule_read (uint16_t handle)

  Sets GATT characteristic with the respective handle to the value of a schedule. First sets the characteristic to the name of the first schedule. On the next read it will set the characteristic to the schedule information. This will repeat until there are no more schedules in the requested channel (saved_channel_num).
- esp_err_t decode_ble_time (uint8_t *packet)

  Converts byte packet into UNIX time value and calls set_time function.
- esp_err_t decode_ble_direct (uint8_t *packet)

  Converts byte packet into channel, brightness, and color information. If RGB channel is written, calls set_color function. If normal LED channel is written, calls channel_on function.
- esp_err_t decode_ble_delete (uint8_t *packet)

  Converts byte packet into schedule name and channel. The schedule on the corresponding channel is then deleted using delete_schedule_by_name function.
- esp_err_t decode_ble_schedule_name (uint8_t *packet, uint16_t length)

  Converts byte packet into string and saves value in saved_name.
- esp_err_t decode_ble_schedule (uint8_t *packet)

  Converts byte packet in order to create schedule object. Recieves name from saved_name.
- uint32_t get_Int32 (uint8_t *begin)

  Takes byte array and converts into a uint32 using the first 4 indices. Assumes Big-endian.

Variables

- uint8_t schedule_value [20]
- uint8_t saved_channel_num
- char schedule_name [20]
- List ∗ schedules [NUM_CHANNELS]

  The linked list associated with each channel. Can be used directly instead of scheduler functions if so desired.
- List ∗ it
- char saved_name [21]
- uint16_t saved_len
- uint8_t state =4
- uint8_t done_string [10] ="Done Read"

5.20.1 Detailed Description

This file contains functions used to decode/set byte packets from GATT server.

Author

Hunaid Puri (hunaid14@gmail.com)

Version

0.1

Date

2020-04-26

Copyright

Copyright (c) 2020

5.20.2 Function Documentation

5.20.2.1 decode_ble_delete()

```
esp_err_t decode_ble_delete (
            uint8_t * packet )
```

Converts byte packet into schedule name and channel. The schedule on the corresponding channel is then deleted using delete_schedule_by_name function.

Parameters

packet	recieved byte packet with schedule name to delete

Returns

esp_err_t

5.20.2.2 decode_ble_direct()

```
esp_err_t decode_ble_direct (
            uint8_t * packet )
```

Converts byte packet into channel, brightness, and color information. If RGB channel is written, calls set_color function. If normal LED channel is written, calls channel_on function.

Parameters

packet	recieved byte packet with led control information

Returns

esp_err_t

5.20.2.3 decode_ble_schedule()

```
esp_err_t decode_ble_schedule (
            uint8_t * packet )
```

Converts byte packet in order to create schedule object. Recieves name from saved_name.

Parameters

packet	recieved byte packet with schedule information

Returns

esp_err_t

5.20.2.4 decode_ble_schedule_name()

```
esp_err_t decode_ble_schedule_name (
            uint8_t * packet,
            uint16_t length )
```

Converts byte packet into string and saves value in saved_name.

Parameters

packet	recieved byte packet with scheudle name
length	length of recieved packet

Returns

esp_err_t

5.20.2.5 decode_ble_time()

```
esp_err_t decode_ble_time (
            uint8_t * packet )
```

Converts byte packet into UNIX time value and calls set_time function.

Parameters

packet	recieved byte packet with time information

Returns

esp_err_t

5.20.2.6 get_Int32()

```
uint32_t get_Int32 (
            uint8_t * begin )
```

Takes byte array and converts into a uint32 using the first 4 indices. Assumes Big-endian.

Parameters

begin	byte array

Returns

uint32_t

5.20.2.7 set_schedule_read()

```
esp_err_t set_schedule_read (
            uint16_t handle )
```

Sets GATT characteristic with the respective handle to the value of a schedule. First sets the characteristic to the name of the first schedule. On the next read it will set the characteristic to the schedule information. This will repeat until there are no more schedules in the requested channel (saved_channel_num).

Parameters

handle	handle for the characteristic that will be read

Returns

esp_err_t

5.20.2.8 start_schedule_read()

```
esp_err_t start_schedule_read (
            uint8_t * packet,
            uint16_t handle )
```

Looks for "Begin Read" from the packet recieved. Saves channel number found in packet in saved_channel_num. Calls set_schedule read, passing handle parameter.

Parameters

packet	recieved byte packet
handle	handle for the characteristic that will be read

Returns

esp_err_t

5.20.3 Variable Documentation

5.20.3.1 done_string

```
uint8_t done_string[10] ="Done Read"
```

5.20.3.2 it

```
List* it
```

5.20.3.3 saved_channel_num

```
uint8_t saved_channel_num
```

5.20.3.4 saved_len

```
uint16_t saved_len
```

5.20.3.5 saved_name

```
char saved_name[21]
```

5.20.3.6 schedule_name

```
char schedule_name[20]
```

5.20.3.7 schedule_value

```
uint8_t schedule_value[20]
```

5.20.3.8 schedules

```
List* schedules[NUM_CHANNELS]
```

The linked list associated with each channel. Can be used directly instead of scheduler functions if so desired.

5.20.3.9 state

```
uint8_t state =4
```

5.21 src/espsntp.cpp File Reference

```
#include "espsntp.h"
```

Functions

- void set_time (uint32_t time)
- void obtain_time (void)

Variables

- const int CONNECTED_BIT = BIT0

5.21.1 Detailed Description

Description: Contains function definitions to start the process of obtaining time from an SNTP server. NOTE: not tested with latest version of project. Was initially developed very early in project and abandoned for other priorites.

Author

: Jesse Cannon (based on https://github.com/espressif/esp-idf/blob/5aa21584cfb4cfe4f2c7f625
_example_main.c)

Date

last modified: 4/26/2020

5.21.2 Function Documentation

5.21.2.1 obtain_time()

```
void obtain_time (
             void  )
```

5.21.2.2 set_time()

```
void set_time (
            uint32_t time )
```

5.21.3 Variable Documentation

5.21.3.1 CONNECTED_BIT

```
const int CONNECTED_BIT = BIT0
```

5.22 src/http.cpp File Reference

File for hosting http server specific to LED controller/scheduler.

```
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event_loop.h"
#include "esp_log.h"
#include "esp_http_server.h"
#include "http.h"
#include "scheduler.h"
#include "espsntp.h"
```

Macros

- #define DELIMITER (";")

Functions

- esp_err_t init_http (httpd_handle_t server)

  Initializes a http server with other handlers defined the file.
- esp_err_t homepage_handler (httpd_req_t *req)

  Handler to retrieve home/direct control page. Accessed when server receives "/" or "/index.html".
- esp_err_t styles_handler (httpd_req_t *req)

  Handler to retrieve styles CSS page. Accessed when server receives "/styles.css".
- esp_err_t schedules_handler (httpd_req_t *req)

  Handler to retrieve schedules page. Accessed when server receives "/schedules.html".
- esp_err_t scripts_handler (httpd_req_t *req)

  Handler to retrieve scripts page. Accessed when server receives "/schedules.js".

- esp_err_t schedule_post_handler (httpd_req_t *req)

  Handler to create new schedule. Accessed when server receives or "/post_sch".
- void schTokenProcess (char *str)

  Function to process ";"-delimited string. Parses out scheduing data and for schedule post handler.
- esp_err_t favicon_ico_get_handler (httpd_req_t *req)

  Handler to retrieve favicon. Accessed when server receives "/favicon.ico" - DOES NOT WORK.
- esp_err_t time_post_handler (httpd_req_t *req)

  Handler to post time to esp32. Accessed when server receives "/time".
- esp_err_t direct_control_post_handler (httpd_req_t *req)

  Handler for direct control. Access when server receives "/direct_control".
- esp_err_t sch_data_post_handler (httpd_req_t *req)

  Handler to retrieve schedule data Access when server receives "/sch_data".

5.22.1 Detailed Description

File for hosting http server specific to LED controller/scheduler.

Author

Andy Yang (andyyeung123@gmail.com)

Version

0.1

Date

2020-04-27

Copyright

Copyright (c) 2020

Description:

Author

: Andy Yeung

Date

last modified: 4/26/2020

5.22.2 Macro Definition Documentation

5.22.2.1 DELIMITER

```
#define DELIMITER (";";")
```

5.22.3 Function Documentation

5.22.3.1 direct_control_post_handler()

```
esp_err_t direct_control_post_handler (
            httpd_req_t * req )
```

Handler for direct control. Access when server receives "/direct_control".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.22.3.2 favicon_ico_get_handler()

```
esp_err_t favicon_ico_get_handler (
            httpd_req_t * req )
```

Handler to retrieve favicon. Accessed when server receives "/favicon.ico" - DOES NOT WORK.

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.22.3.3 homepage_handler()

```
esp_err_t homepage_handler (
            httpd_req_t * req )
```

Handler to retrieve home/direct control page. Accessed when server receives "/" or "/index.html".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.22.3.4 init_http()

```
esp_err_t init_http (
            httpd_handle_t server )
```

Initializes a http server with other handlers defined the file.

Parameters

server	A handle for the server.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.22.3.5 sch_data_post_handler()

```
esp_err_t sch_data_post_handler (
            httpd_req_t * req )
```

Handler to retrieve schedule data Access when server receives "/sch_data".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.22.3.6 schedule_post_handler()

```
esp_err_t schedule_post_handler (
            httpd_req_t * req )
```

Handler to create new schedule. Accessed when server receives or "/post_sch".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.22.3.7 schedules_handler()

```
esp_err_t schedules_handler (
            httpd_req_t * req )
```

Handler to retrieve schedules page. Accessed when server receives "/schedules.html".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.22.3.8 schTokenProcess()

```
void schTokenProcess (
            char * str )
```

Function to process ";"-delimited string. Parses out scheduing data and for schedule post handler.

Parameters

str	Received string

5.22.3.9 scripts_handler()

```
esp_err_t scripts_handler (
            httpd_req_t * req )
```

Handler to retrieve scripts page. Accessed when server receives "/schedules.js".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.22.3.10 styles_handler()

```
esp_err_t styles_handler (
            httpd_req_t * req )
```

Handler to retrieve styles CSS page. Accessed when server receives "/styles.css".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.22.3.11 time_post_handler()

```
esp_err_t time_post_handler (
            httpd_req_t * req )
```

Handler to post time to esp32. Accessed when server receives "/time".

Parameters

req	An incoming HTTP request.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.23 src/led.cpp File Reference

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

Classes

- struct channel

Functions

- esp_err_t shutdown_outputs (void)

  Shuts down all channels and prevents them from turning on again. Enter a permanent shutdown state until clear_shutdown is called or device is reset.
- esp_err_t clear_shutdown (void)

  Clears a shutdown state, allowing outputs to be controlled again.
- void init_channels (void)

  Initialize channels to set all timer and channel configurations.
- void channel_on (uint8_t index, uint8_t brightness)

  Sets brightness of individual channel to turn that output on.
- void channel_off (uint8_t index)

  Sets brightness of individual channel to 0 to turn off channel output.
- void set_color (uint8_t index, uint16_t r, uint16_t g, uint16_t b, uint8_t brightness)

  Controls RGB channel by changing R, G, and B based off chosen brightness.

Variables

- uint8_t shutdown_status = 0

5.23.1 Detailed Description

Description: Contains function definitions to setup and manipulate GPIO outputs as PWM outputs for controlling LED strips

Author

: Primary: Shipra Vaidya, Secondary: Jesse Cannon (bugfixes and shutdown feature)

Date

last modified: 4/26/2020

5.23.2 Function Documentation

5.23.2.1 channel_off()

```
void channel_off (
            uint8_t index )
```

Sets brightness of individual channel to 0 to turn off channel output.

Parameters

index	Value between 0 and 5 corresponding to channel being controlled.

5.23.2.2 channel_on()

```
void channel_on (
            uint8_t index,
            uint8_t brightness )
```

Sets brightness of individual channel to turn that output on.

Parameters

index	Value between 0 and 5 corresponding to channel being controlled.
brightness	Value of brightness chosen per channel.

5.23.2.3 clear_shutdown()

```
esp_err_t clear_shutdown (
            void  )
```

Clears a shutdown state, allowing outputs to be controlled again.

Returns

esp_err_t ESP_OK on success, ESP_FAIL if not in shutdown state.

5.23.2.4 init_channels()

```
void init_channels (
            void  )
```

Initialize channels to set all timer and channel configurations.

5.23.2.5 set_color()

```
void set_color (
            uint8_t index,
            uint16_t r,
            uint16_t g,
            uint16_t b,
            uint8_t brightness )
```

Controls RGB channel by changing R, G, and B based off chosen brightness.

Parameters

index	Value between 0 and 1 correnspondingto two supported RGB channel outputs repectively.
r	R value of chosen color.
g	G value of chosen color.
b	B value of chosen color.
brightness	Value of brightness chosen for RGB channel.

5.23.2.6 shutdown_outputs()

```
esp_err_t shutdown_outputs (
            void  )
```

Shuts down all channels and prevents them from turning on again. Enter a permanent shutdown state until clear↩ _shutdown is called or device is reset.

Returns

esp_err_t ESP_OK on success, ESP_FAIL if already in shutdown state.

5.23.3 Variable Documentation

5.23.3.1 shutdown_status

```
uint8_t shutdown_status = 0
```

5.24 src/main.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_event_loop.h"
#include "esp_http_server.h"
#include "esp_log.h"
#include "esp_spiffs.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "nvs.h"
#include "nvs_flash.h"
#include "lwip/err.h"
#include "lwip/sys.h"
#include "espsntp.h"
```

```
#include "http.h"
#include "led.h"
#include "scheduler.h"
#include "measurement.h"
#include "memory.h"
#include "wifi.h"
#include "bleSL.h"
#include "decode_bluetooth.h"
```

Functions

- void app_main ()

5.24.1 Detailed Description

Description: Contains code to initialize and start 'IoT LED Controller Device' subsystems

Author

: Jesse Cannon, Hunaid Puri, Shipra Vaidya, Andy Yeung

Date

last modified: 4/26/2020

5.24.2 Function Documentation

5.24.2.1 app_main()

```
void app_main ( )
```

5.25 src/measurement.cpp File Reference

```
#include "driver/gpio.h"
#include "measurement.h"
```

Macros

- #define TAG "OC"

Functions

- uint8_t isCurrentFault ()

  Determines if the device is in shutdown from an overcurrent condition.
- uint8_t isVoltageFault ()

  Determines if the device is in shutdown from an overvoltage condition.
- esp_err_t clearFaults ()

  Clears all faults from the system and allows outputs to be controlled again.
- esp_err_t set_current_level (double ampLimit)

  Set the current level that will trigger a fault.
- esp_err_t set_voltage_level (double voltLimit)

  Set the voltage level that will trigger a fault.
- esp_err_t init_oc (void)

  Inits the overcurrent and overvoltage detection systems. The previously used values will be set as the level, with defaults of 24 Volts and 2 Amps triggering faults.

Variables

- uint8_t currentFault = 0
- uint8_t voltageFault = 0

5.25.1 Detailed Description

Description: Contains function definitions to set up and detect overvoltage and overcurrent conditions. NOTE: This is basically just out outline. Needs significant testing with actual hardware before development can finish.

Author

: Jesse Cannon

Date

last modified: 4/26/2020

5.25.2 Macro Definition Documentation

5.25.2.1 TAG

```
#define TAG "OC"
```

5.25.3 Function Documentation

5.25.3.1 clearFaults()

```
esp_err_t clearFaults ( )
```

Clears all faults from the system and allows outputs to be controlled again.

Returns

esp_err_t ESP_OK if successfully re-enables outputs. ESP_FAIL if error occurs.

5.25.3.2 init_oc()

```
esp_err_t init_oc (
            void  )
```

Inits the overcurrent and overvoltage detection systems. The previously used values will be set as the level, with defaults of 24 Volts and 2 Amps triggering faults.

Returns

esp_err_t

5.25.3.3 isCurrentFault()

```
uint8_t isCurrentFault ( )
```

Determines if the device is in shutdown from an overcurrent condition.

Returns

uint8_t A value of 1 means that there is currently a fault. A value of 0 means there is no fault

5.25.3.4 isVoltageFault()

```
uint8_t isVoltageFault ( )
```

Determines if the device is in shutdown from an overvoltage condition.

Returns

uint8_t A value of 1 means that there is currently a fault. A value of 0 means there is no fault

5.25.3.5 set_current_level()

```
esp_err_t set_current_level (
            double ampLimit )
```

Set the current level that will trigger a fault.

Parameters

ampLimit	The current level to be used to determine faults, such as 1.1 Amps.

Returns

esp_err_t ESP_OK if the new level was successfully set, else if error occurred.

5.25.3.6 set_voltage_level()

```
esp_err_t set_voltage_level (
            double voltLimit )
```

Set the voltage level that will trigger a fault.

Parameters

voltLimit	The voltage level to be used to determine faults, such as 12.5 Volts.

Returns

esp_err_t ESP_OK if the new level was successfully set, else if error occurred.

5.25.4 Variable Documentation

5.25.4.1 currentFault

```
uint8_t currentFault = 0
```

5.25.4.2 voltageFault

```
uint8_t voltageFault = 0
```

5.26 src/memory.cpp File Reference

```
#include "memory.h"
```

Functions

- esp_err_t read_settings_to_buffer (void)
- esp_err_t init_spiffs (void)
- esp_err_t init_memory (void)

  Starts the memory system. Call this before calling any other memory related functions.
- esp_err_t recall_schedules (void)

  Recall schedules from persistent memory. All schedules recalled will now be located in the tespective channel's schedule list.
- esp_err_t store_schedules (void)

  Stores all currently running schedules in persistent memory.
- esp_err_t clear_schedule_data (void)

  Clears all schedules from persistent memory. Does not delete the schedules from the list.
- esp_err_t store_setting_string (const char ∗name, char ∗setting)

  Persistently store a string setting.
- esp_err_t store_setting_int (const char ∗name, int setting)

  Persistently store an integer setting.
- esp_err_t store_setting_byte (const char ∗name, uint8_t setting)

  Persistently store a byte setting.
- esp_err_t store_setting_double (const char ∗name, double setting)

  Persistently store a double setting.
- esp_err_t get_setting_string (const char ∗name, char ∗setting)

  Recall a persistent string setting.
- esp_err_t get_setting_int (const char ∗name, int ∗setting)

  Recall a persistent integer setting.
- esp_err_t get_setting_byte (const char ∗name, uint8_t ∗setting)

  Recall a byte string setting. Uint8_t is equivalent to unsigned char.
- esp_err_t get_setting_double (const char ∗name, double ∗setting)

  Recall a persistent double setting. Exact storage depends on ArduinoJson configuration, but typically a maximum of 9 decimal places with rounding. No trailing zeros are included.
- esp_err_t clear_setting_data (void)

  Clear settings from persistent memory and RAM.

Variables

- bool bSPIFFS = false
- bool readNeeded = true
- char ∗ settingsString

5.26.1 Detailed Description

Description: Contains function definitions and settings used by memory subsystem. Provides interface to store values in persistent memory in the JSON format using ArduinoJson-v6.14.1

Author

: Jesse Cannon

Date

last modified: 4/26/2020

5.26.2 Function Documentation

5.26.2.1 clear_schedule_data()

```
esp_err_t clear_schedule_data (
            void  )
```

Clears all schedules from persistent memory. Does not delete the schedules from the list.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.26.2.2 clear_setting_data()

```
esp_err_t clear_setting_data (
            void  )
```

Clear settings from persistent memory and RAM.

Returns

esp_err_t ESP_OK on success. ESP_FAIL if error occurrs.

5.26.2.3 get_setting_byte()

```
esp_err_t get_setting_byte (
            const char * name,
            uint8_t * setting )
```

Recall a byte string setting. Uint8_t is equivalent to unsigned char.

Parameters

name	The name of the setting to recall.
setting	The output of that setting. Set to NULL if not found, the value if found.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail if not found or an error occurred.

5.26.2.4 get_setting_double()

```
esp_err_t get_setting_double (
            const char * name,
            double * setting )
```

Recall a persistent double setting. Exact storage depends on ArduinoJson configuration, but typically a maximum of 9 decimal places with rounding. No trailing zeros are included.

Parameters

name	The name of the setting to recall.
setting	The output of that setting. Set to NULL if not found, the value if found.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail if not found or an error occurred.

5.26.2.5 get_setting_int()

```
esp_err_t get_setting_int (
            const char * name,
            int * setting )
```

Recall a persistent integer setting.

Parameters

name	The name of the setting to recall.
setting	The output of that setting. Set to NULL if not found, the value if found.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail if not found or an error occurred.

5.26.2.6 get_setting_string()

```
esp_err_t get_setting_string (
            const char * name,
            char * setting )
```

Recall a persistent string setting.

Parameters

name	The name of the setting to recall.
setting	The output of that setting. Set to NULL if not found, the value if found.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail if not found or an error occurred.

5.26.2.7 init_memory()

```
esp_err_t init_memory (
            void  )
```

Starts the memory system. Call this before calling any other memory related functions.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.26.2.8 init_spiffs()

```
esp_err_t init_spiffs (
            void  )
```

5.26.2.9 read_settings_to_buffer()

```
esp_err_t read_settings_to_buffer (
            void  )
```

5.26.2.10 recall_schedules()

```
esp_err_t recall_schedules (
            void  )
```

Recall schedules from persistent memory. All schedules recalled will now be located in the tespective channel's schedule list.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.26.2.11 store_schedules()

```
esp_err_t store_schedules (
            void  )
```

Stores all currently running schedules in persistent memory.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.26.2.12 store_setting_byte()

```
esp_err_t store_setting_byte (
            const char * name,
            uint8_t setting )
```

Persistently store a byte setting.

Parameters

name	The name of the setting to store. If the setting with that name already exists it is overwritten with the new value.
setting	The setting to associate with the given name. Uint8_t is equivalent to an unsigned char.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.26.2.13 store_setting_double()

```
esp_err_t store_setting_double (
            const char * name,
            double setting )
```

Persistently store a double setting.

Parameters

name	The name of the setting to store. If the setting with that name already exists it is overwritten with the new value.
setting	The setting to associate with the given name. Exact storage depends on ArduinoJson configuration, but typically a maximum of 9 decimal places with rounding. No trailing zeros are included

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.26.2.14 store_setting_int()

```
esp_err_t store_setting_int (
            const char * name,
            int setting )
```

Persistently store an integer setting.

Parameters

name	The name of the setting to store. If the setting with that name already exists it is overwritten with the new value.
setting	The setting to associate with the given name.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.26.2.15 store_setting_string()

```
esp_err_t store_setting_string (
            const char * name,
            char * setting )
```

Persistently store a string setting.

Parameters

name	The name of the setting to store. If the setting with that name already exists it is overwritten with the new value.
setting	The setting to associate with the given name.

Returns

esp_err_t Returns ESP_OK on success. ESP_Fail or related if an error occurred.

5.26.3 Variable Documentation

5.26.3.1 bSPIFFS

```
bool bSPIFFS = false
```

5.26.3.2 readNeeded

```
bool readNeeded = true
```

5.26.3.3 settingsString

```
char* settingsString
```

5.27 src/rtc.cpp File Reference

```
#include "rtcdefine.h"
```

Functions

- void rtc_config (void)
- esp_err_t readData (uint32_t addr, uint8_t *out)
- esp_err_t writeData (uint32_t addr, uint8_t data)
- esp_err_t setTime (const struct tm *time)

  Set the time of the external RTC to the specified value.
- esp_err_t getTime (struct tm *outTime)

  Get the current time on the external RTC.
- void RTCHandler (void *pvParms)

  The RTOS task that handles RTC related actions. This task syncrhonizes the esp32's and RTC's time.
- esp_err_t ST_StartRTCHandler (void)

  Starts the RTOS task that handles RTC and esp32 time synchronization.

Variables

- spi_device_handle_t rtc

5.27.1 Detailed Description

Description: Provides function definitions to start an RTOS task and interface with an external RTC (MCP79510) Part of the timing subsystem

Author

: Primary: Shipra Vaidya, Secondary: Jesse Cannon

Date

last modified: 4/26/2020

5.27.2 Function Documentation

5.27.2.1 getTime()

```
esp_err_t getTime (
            struct tm * outTime )
```

Get the current time on the external RTC.

Parameters

outTime	The current time on the RTC will be stored in this value. Check the return type against ESP_OK to ensure this value was set correctly.

Returns

esp_err_t ESP_OK on success, else on failure.

5.27.2.2 readData()

```
esp_err_t readData (
            uint32_t addr,
            uint8_t * out )
```

5.27.2.3 rtc_config()

```
void rtc_config (
            void  )
```

5.27.2.4 RTCHandler()

```
void RTCHandler (
            void * pvParms )
```

The RTOS task that handles RTC related actions. This task syncrhonizes the esp32's and RTC's time.

Parameters

pvParms	Required parameter of RTOS tasks. Not used in this task.

5.27.2.5 setTime()

```
esp_err_t setTime (
            const struct tm * time )
```

Set the time of the external RTC to the specified value.

Parameters

time	The time to be set on the device.

Returns

esp_err_t ESP_OK on success, else on failure.

5.27.2.6 ST_StartRTCHandler()

```
esp_err_t ST_StartRTCHandler (
            void  )
```

Starts the RTOS task that handles RTC and esp32 time synchronization.

Returns

esp_err_t ESP_OK if task started successfully, else on failure.

5.27.2.7 writeData()

```
esp_err_t writeData (
            uint32_t addr,
            uint8_t data )
```

5.27.3 Variable Documentation

5.27.3.1 rtc

```
spi_device_handle_t rtc
```

5.28 src/scheduler.cpp File Reference

```
#include "scheduler.h"
```

Macros

- #define MIN(a, b) a < b ? a : b
- #define MAX(a, b) a > b ? a : b

Functions

- esp_err_t create_schedule (uint8_t channel, schedule_object s)

  Create a schedule to associate with the specified channel. If the schedule already exists then it is replaced with the new schedule.
- esp_err_t delete_schedule_by_id (uint8_t channel, uint8_t ID)

  Delete a schedule by ID.
- esp_err_t delete_schedule_by_name (uint8_t channel, char *name)

  Delete a schedule by name.
- esp_err_t disable_schedule_by_id (uint8_t channel, uint8_t ID)

  Disable a schedule by ID. The schedule remains in the linked list but does not run or perform updates.
- esp_err_t disable_schedule_by_name (uint8_t channel, char *name)

  Disable a schedule by name. The schedule remains in the linked list but does not run or perform updates.
- esp_err_t enable_schedule_by_id (uint8_t channel, uint8_t ID)

  Enable a schedule by ID. If the schedule was disabled, the schedule now runs and performs updates.
- esp_err_t enable_schedule_by_name (uint8_t channel, char *name)

  Enable a schedule by name. If the schedule was disabled, the schedule now runs and performs updates.
- esp_err_t disable_all_schedules (void)

  Disables all schedules on all channels. The schedules remain located in memory but no longer run or perform updates.
- esp_err_t enable_all_schedules (void)

  Enable all schedules on all channels. If the schedule was disabled, the schedule now runs and performs updates.
- esp_err_t delete_all_schedules (void)

  Delete all schedules on all channels.
- esp_err_t get_schedule_names (uint8_t channel, char *&out)

  Get a json string of all schedules and their enabled status for a given channel.
- esp_err_t get_schedule (uint8_t channel, char *name, schedule_object *out)

  Get the schedule object specified.
- void update_start_time (schedule_object *s, time_t curr)
- void init_schedule (void)

  Init the scheduler code. Starts the background task if not already started. If this is not called the schedules will not run.

Variables

- List * schedules [NUM_CHANNELS]

  The linked list associated with each channel. Can be used directly instead of scheduler functions if so desired.

5.28.1 Detailed Description

Description: Provides function definitions for running and maintaining repeating schedules as part of the scheduler subsystem.

Author

: Jesse Cannon

Date

last modified: 4/26/2020

5.28.2 Macro Definition Documentation

5.28.2.1 MAX

```
#define MAX(
              a,
              b ) a > b ?  a :  b
```

5.28.2.2 MIN

```
#define MIN(
              a,
              b ) a < b ?  a :  b
```

5.28.3 Function Documentation

5.28.3.1 create_schedule()

```
esp_err_t create_schedule (
             uint8_t channel,
             schedule_object s )
```

Create a schedule to associate with the specified channel. If the schedule already exists then it is replaced with the new schedule.

Parameters

channel	The channel that the schedule will be placed on.
s	The schedule object that will be placed.

Returns

esp_err_t Returns ESP_OK on success. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ER↩ R_NO_MEM if unable to allocate schedule.

5.28.3.2 delete_all_schedules()

```
esp_err_t delete_all_schedules (
            void  )
```

Delete all schedules on all channels.

Returns

esp_err_t ESP_OK on successful deletion.

5.28.3.3 delete_schedule_by_id()

```
esp_err_t delete_schedule_by_id (
            uint8_t channel,
            uint8_t ID )
```

Delete a schedule by ID.

Parameters

channel	The channel the schedule is located on.
ID	The ID of the schedule to delete.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.28.3.4 delete_schedule_by_name()

```
esp_err_t delete_schedule_by_name (
            uint8_t channel,
            char * name )
```

Delete a schedule by name.

Parameters

channel	The channel the schedule is located on.
name	The name of the schedule to delete.

Returns

esp_err_t Returns ESP_OK on successful deletion. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_ERR_NOT_FOUND if schedule not found

5.28.3.5 disable_all_schedules()

```
esp_err_t disable_all_schedules (
            void  )
```

Disables all schedules on all channels. The schedules remain located in memory but no longer run or perform updates.

Returns

esp_err_t ESP_OK on successful disable.

5.28.3.6 disable_schedule_by_id()

```
esp_err_t disable_schedule_by_id (
            uint8_t channel,
            uint8_t ID )
```

Disable a schedule by ID. The schedule remains in the linked list but does not run or perform updates.

Parameters

channel	The channel the schedule is located on.
ID	The name of the schedule to disable.

Returns

esp_err_t ESP_OK on successful disable. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_E↩ RR_NOT_FOUND if schedule not found

5.28.3.7 disable_schedule_by_name()

```
esp_err_t disable_schedule_by_name (
```

```
            uint8_t channel,
            char * name )
```

Disable a schedule by name. The schedule remains in the linked list but does not run or perform updates.

Parameters

channel	The channel the schedule is located on.
name	The name of the schedule to disable.

Returns

esp_err_t ESP_OK on successful disable. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_E↩ RR_NOT_FOUND if schedule not found

5.28.3.8 enable_all_schedules()

```
esp_err_t enable_all_schedules (
            void  )
```

Enable all schedules on all channels. If the schedule was disabled, the schedule now runs and performs updates.

Returns

esp_err_t ESP_OK on successful disable.

5.28.3.9 enable_schedule_by_id()

```
esp_err_t enable_schedule_by_id (
            uint8_t channel,
            uint8_t ID )
```

Enable a schedule by ID. If the schedule was disabled, the schedule now runs and performs updates.

Parameters

channel	The channel the schedule is located on.
ID	The ID of the schedule to enable.

Returns

esp_err_t ESP_OK on successful enable. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_E↩ RR_NOT_FOUND if schedule not found

5.28.3.10 enable_schedule_by_name()

```
esp_err_t enable_schedule_by_name (
            uint8_t channel,
            char * name )
```

Enable a schedule by name. If the schedule was disabled, the schedule now runs and performs updates.

Parameters

channel	The channel the schedule is located on.
name	The name of the schedule to enable.

Returns

esp_err_t ESP_OK on successful enable. ESP_ERR_INVALID_ARG if channel number is invalid. ESP_E↩ RR_NOT_FOUND if schedule not found

5.28.3.11 get_schedule()

```
esp_err_t get_schedule (
            uint8_t channel,
            char * name,
            schedule_object * out )
```

Get the schedule object specified.

Parameters

channel	The channel the schedule is located on.
name	The name of the schedule to find.
out	The schedule returned. NULL if failure occurred.

Returns

esp_err_t ESP_OK if successful. ESP_ERR_INVALID_ARG if channel is invalid. ESP_ERR_NOT_FOUND if the schedule could not be found.

5.28.3.12 get_schedule_names()

```
esp_err_t get_schedule_names (
            uint8_t channel,
            char *& out )
```

Get a json string of all schedules and their enabled status for a given channel.

Parameters

channel	The channel to get the schedule names and status from.
out	The json string that is returned. Format is '{ "Name1":0, "Name2":1, "Name3":0 }'

Returns

esp_err_t ESP_OK if successful. ESP_ERR_INVALID_ARG if channel is invalid.

5.28.3.13 init_schedule()

```
void init_schedule (
            void  )
```

Init the scheduler code. Starts the background task if not already started. If this is not called the schedules will not run.

5.28.3.14 update_start_time()

```
void update_start_time (
            schedule_object * s,
            time_t curr )
```

5.28.4 Variable Documentation

5.28.4.1 schedules

```
List* schedules[NUM_CHANNELS]
```

The linked list associated with each channel. Can be used directly instead of scheduler functions if so desired.

5.29 src/sdkconfig.h File Reference

Macros

- #define CONFIG_ENABLE_ARDUINO_DEPENDS 1
- #define CONFIG_AUTOSTART_ARDUINO 1
- #define CONFIG_ARDUINO_RUNNING_CORE 1
- #define CONFIG_ARDUINO_UDP_RUN_CORE1 1
- #define CONFIG_ARDUINO_EVENT_RUN_CORE1 1
- #define CONFIG_ARDUINO_EVENT_RUNNING_CORE 1
- #define CONFIG_ARDUINO_UDP_RUNNING_CORE 1
- #define CONFIG_GATTC_ENABLE 1
- #define CONFIG_ESP32_PHY_MAX_TX_POWER 20
- #define CONFIG_TRACEMEM_RESERVE_DRAM 0x0
- #define CONFIG_FREERTOS_MAX_TASK_NAME_LEN 16
- #define CONFIG_MQTT_TRANSPORT_SSL 1
- #define CONFIG_BLE_SMP_ENABLE 1
- #define CONFIG_FATFS_LFN_NONE 1
- #define CONFIG_SDP_INITIAL_TRACE_LEVEL 2
- #define CONFIG_MB_SERIAL_TASK_PRIO 10
- #define CONFIG_MQTT_PROTOCOL_311 1
- #define CONFIG_TCP_RECVMBOX_SIZE 6
- #define CONFIG_FATFS_CODEPAGE_437 1
- #define CONFIG_BLE_SCAN_DUPLICATE 1
- #define CONFIG_AVDT_TRACE_LEVEL_WARNING 1
- #define CONFIG_AWS_IOT_SHADOW_MAX_SIMULTANEOUS_ACKS 10
- #define CONFIG_TCP_WND_DEFAULT 5744
- #define CONFIG_PARTITION_TABLE_OFFSET 0x8000
- #define CONFIG_SW_COEXIST_ENABLE 1
- #define CONFIG_SPIFFS_USE_MAGIC_LENGTH 1
- #define CONFIG_AVCT_INITIAL_TRACE_LEVEL 2
- #define CONFIG_IPC_TASK_STACK_SIZE 1024
- #define CONFIG_WIFI_PROV_SCAN_MAX_ENTRIES 16
- #define CONFIG_FATFS_PER_FILE_CACHE 1
- #define CONFIG_ESPTOOLPY_FLASHFREQ "40m"
- #define CONFIG_AWS_IOT_SHADOW_MAX_SIZE_OF_THING_NAME 20
- #define CONFIG_MBEDTLS_KEY_EXCHANGE_RSA 1
- #define CONFIG_UDP_RECVMBOX_SIZE 6
- #define CONFIG_SPI_FLASH_YIELD_DURING_ERASE 1
- #define CONFIG_FREERTOS_QUEUE_REGISTRY_SIZE 0
- #define CONFIG_MBEDTLS_AES_C 1
- #define CONFIG_MBEDTLS_ECP_DP_SECP521R1_ENABLED 1
- #define CONFIG_ESP32_WIFI_SOFTAP_BEACON_MAX_LEN 752
- #define CONFIG_MBEDTLS_GCM_C 1
- #define CONFIG_ESPTOOLPY_FLASHSIZE "2MB"
- #define CONFIG_HEAP_POISONING_DISABLED 1
- #define CONFIG_SPIFFS_CACHE_WR 1
- #define CONFIG_BROWNOUT_DET_LVL_SEL_0 1
- #define CONFIG_ESP32_WIFI_DYNAMIC_TX_BUFFER 1
- #define CONFIG_SPIFFS_CACHE 1
- #define CONFIG_INT_WDT 1
- #define CONFIG_BTDM_CONTROLLER_BLE_MAX_CONN 3
- #define CONFIG_MBEDTLS_SSL_PROTO_TLS1 1
- #define CONFIG_ESP_GRATUITOUS_ARP 1

- #define CONFIG_AWS_IOT_SHADOW_MAX_SIZE_OF_UNIQUE_CLIENT_ID_BYTES 80
- #define CONFIG_MBEDTLS_ECDSA_C 1
- #define CONFIG_ESPTOOLPY_FLASHFREQ_40M 1
- #define CONFIG_LOG_BOOTLOADER_LEVEL_INFO 1
- #define CONFIG_ESPTOOLPY_FLASHSIZE_2MB 1
- #define CONFIG_HTTPD_MAX_REQ_HDR_LEN 512
- #define CONFIG_BTDM_CONTROLLER_PINNED_TO_CORE 0
- #define CONFIG_AWS_IOT_MQTT_PORT 8883
- #define CONFIG_FREERTOS_THREAD_LOCAL_STORAGE_POINTERS 1
- #define CONFIG_MBEDTLS_ECDH_C 1
- #define CONFIG_MBEDTLS_KEY_EXCHANGE_ELLIPTIC_CURVE 1
- #define CONFIG_ESP32_WIFI_STATIC_RX_BUFFER_NUM 10
- #define CONFIG_AWS_IOT_MQTT_MAX_RECONNECT_WAIT_INTERVAL 128000
- #define CONFIG_MBEDTLS_SSL_ALPN 1
- #define CONFIG_BTM_TRACE_LEVEL_WARNING 1
- #define CONFIG_MBEDTLS_PEM_WRITE_C 1
- #define CONFIG_RFCOMM_TRACE_LEVEL_WARNING 1
- #define CONFIG_LOG_DEFAULT_LEVEL_INFO 1
- #define CONFIG_BT_RESERVE_DRAM 0xdb5c
- #define CONFIG_APP_COMPILE_TIME_DATE 1
- #define CONFIG_FATFS_FS_LOCK 0
- #define CONFIG_IP_LOST_TIMER_INTERVAL 120
- #define CONFIG_SPIFFS_META_LENGTH 4
- #define CONFIG_ESP32_PANIC_PRINT_REBOOT 1
- #define CONFIG_MB_CONTROLLER_NOTIFY_QUEUE_SIZE 20
- #define CONFIG_MBEDTLS_ECP_DP_BP384R1_ENABLED 1
- #define CONFIG_MBEDTLS_ECP_DP_SECP256K1_ENABLED 1
- #define CONFIG_AWS_IOT_MQTT_RX_BUF_LEN 512
- #define CONFIG_MB_SERIAL_BUF_SIZE 256
- #define CONFIG_CONSOLE_UART_BAUDRATE 115200
- #define CONFIG_LWIP_MAX_SOCKETS 10
- #define CONFIG_LWIP_NETIF_LOOPBACK 1
- #define CONFIG_MCA_TRACE_LEVEL_WARNING 1
- #define CONFIG_ESP32_PTHREAD_TASK_NAME_DEFAULT "pthread"
- #define CONFIG_EMAC_TASK_PRIORITY 20
- #define CONFIG_TIMER_TASK_STACK_DEPTH 2048
- #define CONFIG_TCP_MSS 1436
- #define CONFIG_MBEDTLS_ECP_DP_CURVE25519_ENABLED 1
- #define CONFIG_BTIF_INITIAL_TRACE_LEVEL 2
- #define CONFIG_BTDM_CONTROLLER_BLE_MAX_CONN_EFF 3
- #define CONFIG_EFUSE_CODE_SCHEME_COMPAT_3_4 1
- #define CONFIG_FATFS_CODEPAGE 437
- #define CONFIG_APPL_TRACE_LEVEL_WARNING 1
- #define CONFIG_BTC_INITIAL_TRACE_LEVEL 2
- #define CONFIG_ESP32_DEFAULT_CPU_FREQ_160 1
- #define CONFIG_ULP_COPROC_RESERVE_MEM 0
- #define CONFIG_LWIP_MAX_UDP_PCBS 16
- #define CONFIG_ESPTOOLPY_BAUD 115200
- #define CONFIG_INT_WDT_CHECK_CPU1 1
- #define CONFIG_AVRC_INITIAL_TRACE_LEVEL 2
- #define CONFIG_ADC_CAL_LUT_ENABLE 1
- #define CONFIG_AWS_IOT_MQTT_TX_BUF_LEN 512
- #define CONFIG_FLASHMODE_DIO 1
- #define CONFIG_ESPTOOLPY_AFTER_RESET 1
- #define CONFIG_OPTIMIZATION_ASSERTIONS_ENABLED 1

- #define CONFIG_LWIP_DHCPS_MAX_STATION_NUM 8
- #define CONFIG_TOOLPREFIX "xtensa-esp32-elf-"
- #define CONFIG_MBEDTLS_ECP_C 1
- #define CONFIG_FREERTOS_IDLE_TASK_STACKSIZE 1536
- #define CONFIG_MBEDTLS_RC4_DISABLED 1
- #define CONFIG_GAP_TRACE_LEVEL_WARNING 1
- #define CONFIG_CONSOLE_UART_NUM 0
- #define CONFIG_AWS_IOT_SHADOW_MAX_JSON_TOKEN_EXPECTED 120
- #define CONFIG_ESP32_APPTRACE_LOCK_ENABLE 1
- #define CONFIG_PTHREAD_STACK_MIN 768
- #define CONFIG_ESP32_RTC_CLOCK_SOURCE_INTERNAL_RC 1
- #define CONFIG_ESPTOOLPY_BAUD_115200B 1
- #define CONFIG_TCP_OVERSIZE_MSS 1
- #define CONFIG_FOUR_UNIVERSAL_MAC_ADDRESS 1
- #define CONFIG_CONSOLE_UART_DEFAULT 1
- #define CONFIG_MBEDTLS_SSL_MAX_CONTENT_LEN 16384
- #define CONFIG_NUMBER_OF_UNIVERSAL_MAC_ADDRESS 4
- #define CONFIG_GATT_TRACE_LEVEL_WARNING 1
- #define CONFIG_ESPTOOLPY_FLASHSIZE_DETECT 1
- #define CONFIG_TIMER_TASK_STACK_SIZE 3584
- #define CONFIG_BTIF_TRACE_LEVEL_WARNING 1
- #define CONFIG_ESP32_ENABLE_COREDUMP_TO_NONE 1
- #define CONFIG_HCI_INITIAL_TRACE_LEVEL 2
- #define CONFIG_AVDT_INITIAL_TRACE_LEVEL 2
- #define CONFIG_MBEDTLS_X509_CRL_PARSE_C 1
- #define CONFIG_FREERTOS_CHECK_MUTEX_GIVEN_BY_OWNER 1
- #define CONFIG_HTTPD_PURGE_BUF_LEN 32
- #define CONFIG_SCAN_DUPLICATE_BY_DEVICE_ADDR 1
- #define CONFIG_AWS_IOT_SHADOW_MAX_SHADOW_TOPIC_LENGTH_WITHOUT_THINGNAME 60
- #define CONFIG_UNITY_ENABLE_IDF_TEST_RUNNER 1
- #define CONFIG_MB_SERIAL_TASK_STACK_SIZE 2048
- #define CONFIG_GATTS_SEND_SERVICE_CHANGE_AUTO 1
- #define CONFIG_LWIP_DHCPS_LEASE_UNIT 60
- #define CONFIG_EFUSE_MAX_BLK_LEN 192
- #define CONFIG_SPIFFS_USE_MAGIC 1
- #define CONFIG_TCPIP_TASK_STACK_SIZE 2048
- #define CONFIG_BLUFI_TRACE_LEVEL_WARNING 1
- #define CONFIG_BLUEDROID_PINNED_TO_CORE_0 1
- #define CONFIG_TASK_WDT 1
- #define CONFIG_RFCOMM_INITIAL_TRACE_LEVEL 2
- #define CONFIG_MAIN_TASK_STACK_SIZE 3584
- #define CONFIG_SPIFFS_PAGE_CHECK 1
- #define CONFIG_ESP32_WIFI_TASK_PINNED_TO_CORE_0 1
- #define CONFIG_LWIP_MAX_ACTIVE_TCP 16
- #define CONFIG_TASK_WDT_TIMEOUT_S 5
- #define CONFIG_INT_WDT_TIMEOUT_MS 300
- #define CONFIG_ESPTOOLPY_FLASHMODE "dio"
- #define CONFIG_BTC_TASK_STACK_SIZE 3072
- #define CONFIG_BLUEDROID_ENABLED 1
- #define CONFIG_NEWLIB_STDIN_LINE_ENDING_CR 1
- #define CONFIG_MBEDTLS_KEY_EXCHANGE_ECDH_RSA 1
- #define CONFIG_ESPTOOLPY_BEFORE "default_reset"
- #define CONFIG_ADC2_DISABLE_DAC 1
- #define CONFIG_BLE_ADV_REPORT_FLOW_CONTROL_NUM 100
- #define CONFIG_ESP32_REV_MIN_0 1

- #define CONFIG_LOG_DEFAULT_LEVEL 3
- #define CONFIG_FREERTOS_ASSERT_ON_UNTESTED_FUNCTION 1
- #define CONFIG_TIMER_QUEUE_LENGTH 10
- #define CONFIG_ESP32_REV_MIN 0
- #define CONFIG_SUPPRESS_SELECT_DEBUG_OUTPUT 1
- #define CONFIG_GATTS_SEND_SERVICE_CHANGE_MODE 0
- #define CONFIG_TCPIP_TASK_AFFINITY_NO_AFFINITY 1
- #define CONFIG_MAKE_WARN_UNDEFINED_VARIABLES 1
- #define CONFIG_FATFS_TIMEOUT_MS 10000
- #define CONFIG_ESP32_WIFI_DYNAMIC_RX_BUFFER_NUM 32
- #define CONFIG_ESP_HTTP_CLIENT_ENABLE_HTTPS 1
- #define CONFIG_PAN_INITIAL_TRACE_LEVEL 2
- #define CONFIG_MBEDTLS_CCM_C 1
- #define CONFIG_SPI_MASTER_ISR_IN_IRAM 1
- #define CONFIG_MCA_INITIAL_TRACE_LEVEL 2
- #define CONFIG_ESP32_PHY_MAX_WIFI_TX_POWER 20
- #define CONFIG_A2D_INITIAL_TRACE_LEVEL 2
- #define CONFIG_ESP32_RTC_CLK_CAL_CYCLES 1024
- #define CONFIG_ESP32_WIFI_TX_BA_WIN 6
- #define CONFIG_ESP32_WIFI_NVS_ENABLED 1
- #define CONFIG_MDNS_MAX_SERVICES 10
- #define CONFIG_IDF_TARGET_ESP32 1
- #define CONFIG_EMAC_CHECK_LINK_PERIOD_MS 2000
- #define CONFIG_MBEDTLS_ECP_DP_SECP224R1_ENABLED 1
- #define CONFIG_SPI_FLASH_ERASE_YIELD_DURATION_MS 20
- #define CONFIG_LIBSODIUM_USE_MBEDTLS_SHA 1
- #define CONFIG_AWS_IOT_SDK 1
- #define CONFIG_DMA_RX_BUF_NUM 10
- #define CONFIG_MBEDTLS_ECP_DP_SECP384R1_ENABLED 1
- #define CONFIG_TCP_SYNMAXRTX 6
- #define CONFIG_MBEDTLS_KEY_EXCHANGE_ECDHE_ECDSA 1
- #define CONFIG_BTDM_CONTROLLER_BR_EDR_MAX_SYNC_CONN_EFF 0
- #define CONFIG_PYTHON "python"
- #define CONFIG_MBEDTLS_ECP_NIST_OPTIM 1
- #define CONFIG_ESP32_TIME_SYSCALL_USE_RTC_FRC1 1
- #define CONFIG_ESPTOOLPY_COMPRESSED 1
- #define CONFIG_PARTITION_TABLE_FILENAME "partitions_singleapp.csv"
- #define CONFIG_MB_CONTROLLER_STACK_SIZE 4096
- #define CONFIG_TCP_SND_BUF_DEFAULT 5744
- #define CONFIG_GARP_TMR_INTERVAL 60
- #define CONFIG_LWIP_DHCP_MAX_NTP_SERVERS 1
- #define CONFIG_BNEP_INITIAL_TRACE_LEVEL 2
- #define CONFIG_HCI_TRACE_LEVEL_WARNING 1
- #define CONFIG_TCP_MSL 60000
- #define CONFIG_MBEDTLS_SSL_PROTO_TLS1_1 1
- #define CONFIG_LWIP_SO_REUSE_RXTOALL 1
- #define CONFIG_MB_CONTROLLER_NOTIFY_TIMEOUT 20
- #define CONFIG_ESP32_WIFI_MGMT_SBUF_NUM 32
- #define CONFIG_PARTITION_TABLE_SINGLE_APP 1
- #define CONFIG_UNITY_ENABLE_FLOAT 1
- #define CONFIG_ESP32_WIFI_RX_BA_WIN 6
- #define CONFIG_MBEDTLS_X509_CSR_PARSE_C 1
- #define CONFIG_SPIFFS_USE_MTIME 1
- #define CONFIG_BTC_TRACE_LEVEL_WARNING 1
- #define CONFIG_EMAC_TASK_STACK_SIZE 3072

- #define CONFIG_SMP_TRACE_LEVEL_WARNING 1
- #define CONFIG_MB_QUEUE_LENGTH 20
- #define CONFIG_SW_COEXIST_PREFERENCE_VALUE 2
- #define CONFIG_MBEDTLS_KEY_EXCHANGE_ECDHE_RSA 1
- #define CONFIG_LWIP_DHCP_DOES_ARP_CHECK 1
- #define CONFIG_FREERTOS_TASK_FUNCTION_WRAPPER 1
- #define CONFIG_SYSTEM_EVENT_TASK_STACK_SIZE 2304
- #define CONFIG_BOOTLOADER_VDDSDIO_BOOST_1_9V 1
- #define CONFIG_A2D_TRACE_LEVEL_WARNING 1
- #define CONFIG_ESP32_DEEP_SLEEP_WAKEUP_DELAY 2000
- #define CONFIG_AWS_IOT_MQTT_NUM_SUBSCRIBE_HANDLERS 5
- #define CONFIG_BROWNOUT_DET_LVL 0
- #define CONFIG_MBEDTLS_PEM_PARSE_C 1
- #define CONFIG_SPIFFS_GC_MAX_RUNS 10
- #define CONFIG_ESP32_APPTRACE_DEST_NONE 1
- #define CONFIG_MBEDTLS_INTERNAL_MEM_ALLOC 1
- #define CONFIG_MBEDTLS_SSL_PROTO_TLS1_2 1
- #define CONFIG_MBEDTLS_KEY_EXCHANGE_DHE_RSA 1
- #define CONFIG_ESP32_WIFI_DYNAMIC_TX_BUFFER_NUM 32
- #define CONFIG_HTTPD_MAX_URI_LEN 512
- #define CONFIG_MBEDTLS_ECP_DP_BP256R1_ENABLED 1
- #define CONFIG_AVCT_TRACE_LEVEL_WARNING 1
- #define CONFIG_MBEDTLS_ECP_DP_SECP224K1_ENABLED 1
- #define CONFIG_TASK_WDT_CHECK_IDLE_TASK_CPU1 1
- #define CONFIG_ESP32_DEFAULT_CPU_FREQ_MHZ 160
- #define CONFIG_MBEDTLS_HARDWARE_AES 1
- #define CONFIG_FREERTOS_HZ 100
- #define CONFIG_LOG_COLORS 1
- #define CONFIG_OSI_TRACE_LEVEL_WARNING 1
- #define CONFIG_ESP32_PHY_CALIBRATION_AND_DATA_STORAGE 1
- #define CONFIG_STACK_CHECK_NONE 1
- #define CONFIG_ADC_CAL_EFUSE_TP_ENABLE 1
- #define CONFIG_BNEP_TRACE_LEVEL_WARNING 1
- #define CONFIG_FREERTOS_ASSERT_FAIL_ABORT 1
- #define CONFIG_BROWNOUT_DET 1
- #define CONFIG_AWS_IOT_SHADOW_MAX_SIMULTANEOUS_THINGNAMES 10
- #define CONFIG_ESP32_XTAL_FREQ 40
- #define CONFIG_OSI_INITIAL_TRACE_LEVEL 2
- #define CONFIG_MONITOR_BAUD_115200B 1
- #define CONFIG_LOG_BOOTLOADER_LEVEL 3
- #define CONFIG_MBEDTLS_TLS_ENABLED 1
- #define CONFIG_LWIP_MAX_RAW_PCBS 16
- #define CONFIG_BTU_TASK_STACK_SIZE 4096
- #define CONFIG_SMP_ENABLE 1
- #define CONFIG_HID_TRACE_LEVEL_WARNING 1
- #define CONFIG_AVRC_TRACE_LEVEL_WARNING 1
- #define CONFIG_MBEDTLS_SSL_SESSION_TICKETS 1
- #define CONFIG_SPIFFS_MAX_PARTITIONS 3
- #define CONFIG_ESP_ERR_TO_NAME_LOOKUP 1
- #define CONFIG_BTDM_CONTROLLER_PINNED_TO_CORE_0 1
- #define CONFIG_MBEDTLS_SSL_RENEGOTIATION 1
- #define CONFIG_HID_INITIAL_TRACE_LEVEL 2
- #define CONFIG_ESPTOOLPY_BEFORE_RESET 1
- #define CONFIG_MB_EVENT_QUEUE_TIMEOUT 20
- #define CONFIG_ESPTOOLPY_BAUD_OTHER_VAL 115200

- #define CONFIG_SPIFFS_OBJ_NAME_LEN 32
- #define CONFIG_ESP32_PTHREAD_TASK_PRIO_DEFAULT 5
- #define CONFIG_BTDM_CONTROLLER_BR_EDR_MAX_ACL_CONN_EFF 0
- #define CONFIG_PARTITION_TABLE_MD5 1
- #define CONFIG_TCPIP_RECVMBOX_SIZE 32
- #define CONFIG_TCP_MAXRTX 12
- #define CONFIG_BTM_INITIAL_TRACE_LEVEL 2
- #define CONFIG_ESPTOOLPY_AFTER "hard_reset"
- #define CONFIG_TCPIP_TASK_AFFINITY 0x7FFFFFFF
- #define CONFIG_LWIP_SO_REUSE 1
- #define CONFIG_ESP32_XTAL_FREQ_40 1
- #define CONFIG_BTDM_CONTROLLER_MODE_BLE_ONLY 1
- #define CONFIG_DMA_TX_BUF_NUM 10
- #define CONFIG_LWIP_MAX_LISTENING_TCP 16
- #define CONFIG_FREERTOS_INTERRUPT_BACKTRACE 1
- #define CONFIG_WL_SECTOR_SIZE 4096
- #define CONFIG_ESP32_DEBUG_OCDAWARE 1
- #define CONFIG_MQTT_TRANSPORT_WEBSOCKET 1
- #define CONFIG_TIMER_TASK_PRIORITY 1
- #define CONFIG_MBEDTLS_TLS_CLIENT 1
- #define CONFIG_AWS_IOT_MQTT_MIN_RECONNECT_WAIT_INTERVAL 1000
- #define CONFIG_BTDM_CONTROLLER_HCI_MODE_VHCI 1
- #define CONFIG_BT_ENABLED 1
- #define CONFIG_ESP32_DEFAULT_PTHREAD_CORE_NO_AFFINITY 1
- #define CONFIG_SDP_TRACE_LEVEL_WARNING 1
- #define CONFIG_SW_COEXIST_PREFERENCE_BALANCE 1
- #define CONFIG_MBEDTLS_ECP_DP_SECP256R1_ENABLED 1
- #define CONFIG_MONITOR_BAUD 115200
- #define CONFIG_ESP32_PTHREAD_TASK_CORE_DEFAULT -1
- #define CONFIG_ESP32_DEBUG_STUBS_ENABLE 1
- #define CONFIG_BLE_ESTABLISH_LINK_CONNECTION_TIMEOUT 30
- #define CONFIG_TCPIP_LWIP 1
- #define CONFIG_REDUCE_PHY_TX_POWER 1
- #define CONFIG_BOOTLOADER_WDT_TIME_MS 9000
- #define CONFIG_PAN_TRACE_LEVEL_WARNING 1
- #define CONFIG_FREERTOS_CORETIMER_0 1
- #define CONFIG_PARTITION_TABLE_CUSTOM_FILENAME "partitions.csv"
- #define CONFIG_MBEDTLS_HAVE_TIME 1
- #define CONFIG_FREERTOS_CHECK_STACKOVERFLOW_CANARY 1
- #define CONFIG_TCP_QUEUE_OOSEQ 1
- #define CONFIG_GATTS_ENABLE 1
- #define CONFIG_ADC_CAL_EFUSE_VREF_ENABLE 1
- #define CONFIG_MBEDTLS_TLS_SERVER 1
- #define CONFIG_MBEDTLS_TLS_SERVER_AND_CLIENT 1
- #define CONFIG_BLE_ADV_REPORT_FLOW_CONTROL_SUPPORTED 1
- #define CONFIG_FREERTOS_ISR_STACKSIZE 1536
- #define CONFIG_SUPPORT_TERMIOS 1
- #define CONFIG_OPENSSL_ASSERT_DO_NOTHING 1
- #define CONFIG_IDF_TARGET "esp32"
- #define CONFIG_WL_SECTOR_SIZE_4096 1
- #define CONFIG_OPTIMIZATION_LEVEL_DEBUG 1
- #define CONFIG_GATT_INITIAL_TRACE_LEVEL 2
- #define CONFIG_FREERTOS_NO_AFFINITY 0x7FFFFFFF
- #define CONFIG_AWS_IOT_MQTT_HOST ""
- #define CONFIG_L2CAP_TRACE_LEVEL_WARNING 1

- #define CONFIG_ESP32_WIFI_AMPDU_TX_ENABLED 1
- #define CONFIG_HTTPD_ERR_RESP_NO_DELAY 1
- #define CONFIG_MB_TIMER_INDEX 0
- #define CONFIG_SCAN_DUPLICATE_TYPE 0
- #define CONFIG_MBEDTLS_ECP_DP_SECP192R1_ENABLED 1
- #define CONFIG_APPL_INITIAL_TRACE_LEVEL 2
- #define CONFIG_MBEDTLS_ECP_DP_BP512R1_ENABLED 1
- #define CONFIG_SPI_FLASH_ERASE_YIELD_TICKS 1
- #define CONFIG_SMP_INITIAL_TRACE_LEVEL 2
- #define CONFIG_MBEDTLS_KEY_EXCHANGE_ECDH_ECDSA 1
- #define CONFIG_SPI_SLAVE_ISR_IN_IRAM 1
- #define CONFIG_L2CAP_INITIAL_TRACE_LEVEL 2
- #define CONFIG_SYSTEM_EVENT_QUEUE_SIZE 32
- #define CONFIG_BT_ACL_CONNECTIONS 4
- #define CONFIG_ESP32_WIFI_TX_BUFFER_TYPE 1
- #define CONFIG_BOOTLOADER_WDT_ENABLE 1
- #define CONFIG_GAP_INITIAL_TRACE_LEVEL 2
- #define CONFIG_ESP32_WIFI_AMPDU_RX_ENABLED 1
- #define CONFIG_LWIP_LOOPBACK_MAX_PBUFS 8
- #define CONFIG_MB_TIMER_GROUP 0
- #define CONFIG_SPI_FLASH_ROM_DRIVER_PATCH 1
- #define CONFIG_MQTT_TRANSPORT_WEBSOCKET_SECURE 1
- #define CONFIG_SPIFFS_PAGE_SIZE 256
- #define CONFIG_MBEDTLS_ECP_DP_SECP192K1_ENABLED 1
- #define CONFIG_ESP32_DPORT_WORKAROUND 1
- #define CONFIG_TASK_WDT_CHECK_IDLE_TASK_CPU0 1
- #define CONFIG_ESP32_PTHREAD_TASK_STACK_SIZE_DEFAULT 3072
- #define CONFIG_MB_TIMER_PORT_ENABLED 1
- #define CONFIG_DUPLICATE_SCAN_CACHE_SIZE 50
- #define CONFIG_MONITOR_BAUD_OTHER_VAL 115200
- #define CONFIG_NEWLIB_STDOUT_LINE_ENDING_CRLF 1
- #define CONFIG_ESPTOOLPY_PORT "COM19"
- #define CONFIG_SPI_FLASH_WRITING_DANGEROUS_REGIONS_ABORTS 1
- #define CONFIG_UNITY_ENABLE_DOUBLE 1
- #define CONFIG_BLE_ADV_REPORT_DISCARD_THRSHOLD 20
- #define CONFIG_BLUEDROID_PINNED_TO_CORE 0
- #define CONFIG_ESP32_WIFI_IRAM_OPT 1
- #define CONFIG_BLUFI_INITIAL_TRACE_LEVEL 2

5.29.1 Macro Definition Documentation

5.29.1.1 CONFIG_A2D_INITIAL_TRACE_LEVEL

```
#define CONFIG_A2D_INITIAL_TRACE_LEVEL 2
```

5.29.1.2 CONFIG_A2D_TRACE_LEVEL_WARNING

```
#define CONFIG_A2D_TRACE_LEVEL_WARNING 1
```

5.29.1.3 CONFIG_ADC2_DISABLE_DAC

```
#define CONFIG_ADC2_DISABLE_DAC 1
```

5.29.1.4 CONFIG_ADC_CAL_EFUSE_TP_ENABLE

```
#define CONFIG_ADC_CAL_EFUSE_TP_ENABLE 1
```

5.29.1.5 CONFIG_ADC_CAL_EFUSE_VREF_ENABLE

```
#define CONFIG_ADC_CAL_EFUSE_VREF_ENABLE 1
```

5.29.1.6 CONFIG_ADC_CAL_LUT_ENABLE

```
#define CONFIG_ADC_CAL_LUT_ENABLE 1
```

5.29.1.7 CONFIG_APP_COMPILE_TIME_DATE

```
#define CONFIG_APP_COMPILE_TIME_DATE 1
```

5.29.1.8 CONFIG_APPL_INITIAL_TRACE_LEVEL

```
#define CONFIG_APPL_INITIAL_TRACE_LEVEL 2
```

5.29.1.9 CONFIG_APPL_TRACE_LEVEL_WARNING

```
#define CONFIG_APPL_TRACE_LEVEL_WARNING 1
```

5.29.1.10 CONFIG_ARDUINO_EVENT_RUN_CORE1

```
#define CONFIG_ARDUINO_EVENT_RUN_CORE1 1
```

5.29.1.11 CONFIG_ARDUINO_EVENT_RUNNING_CORE

```
#define CONFIG_ARDUINO_EVENT_RUNNING_CORE 1
```

5.29.1.12 CONFIG_ARDUINO_RUNNING_CORE

```
#define CONFIG_ARDUINO_RUNNING_CORE 1
```

5.29.1.13 CONFIG_ARDUINO_UDP_RUN_CORE1

```
#define CONFIG_ARDUINO_UDP_RUN_CORE1 1
```

5.29.1.14 CONFIG_ARDUINO_UDP_RUNNING_CORE

```
#define CONFIG_ARDUINO_UDP_RUNNING_CORE 1
```

5.29.1.15 CONFIG_AUTOSTART_ARDUINO

```
#define CONFIG_AUTOSTART_ARDUINO 1
```

5.29.1.16 CONFIG_AVCT_INITIAL_TRACE_LEVEL

```
#define CONFIG_AVCT_INITIAL_TRACE_LEVEL 2
```

5.29.1.17 CONFIG_AVCT_TRACE_LEVEL_WARNING

```
#define CONFIG_AVCT_TRACE_LEVEL_WARNING 1
```

5.29.1.18 CONFIG_AVDT_INITIAL_TRACE_LEVEL

```
#define CONFIG_AVDT_INITIAL_TRACE_LEVEL 2
```

5.29.1.19 CONFIG_AVDT_TRACE_LEVEL_WARNING

```
#define CONFIG_AVDT_TRACE_LEVEL_WARNING 1
```

5.29.1.20 CONFIG_AVRC_INITIAL_TRACE_LEVEL

```
#define CONFIG_AVRC_INITIAL_TRACE_LEVEL 2
```

5.29.1.21 CONFIG_AVRC_TRACE_LEVEL_WARNING

```
#define CONFIG_AVRC_TRACE_LEVEL_WARNING 1
```

5.29.1.22 CONFIG_AWS_IOT_MQTT_HOST

```
#define CONFIG_AWS_IOT_MQTT_HOST ""
```

5.29.1.23 CONFIG_AWS_IOT_MQTT_MAX_RECONNECT_WAIT_INTERVAL

```
#define CONFIG_AWS_IOT_MQTT_MAX_RECONNECT_WAIT_INTERVAL 128000
```

5.29.1.24 CONFIG_AWS_IOT_MQTT_MIN_RECONNECT_WAIT_INTERVAL

```
#define CONFIG_AWS_IOT_MQTT_MIN_RECONNECT_WAIT_INTERVAL 1000
```

5.29.1.25 CONFIG_AWS_IOT_MQTT_NUM_SUBSCRIBE_HANDLERS

```
#define CONFIG_AWS_IOT_MQTT_NUM_SUBSCRIBE_HANDLERS 5
```

5.29.1.26 CONFIG_AWS_IOT_MQTT_PORT

```
#define CONFIG_AWS_IOT_MQTT_PORT 8883
```

5.29.1.27 CONFIG_AWS_IOT_MQTT_RX_BUF_LEN

```
#define CONFIG_AWS_IOT_MQTT_RX_BUF_LEN 512
```

5.29.1.28 CONFIG_AWS_IOT_MQTT_TX_BUF_LEN

```
#define CONFIG_AWS_IOT_MQTT_TX_BUF_LEN 512
```

5.29.1.29 CONFIG_AWS_IOT_SDK

```
#define CONFIG_AWS_IOT_SDK 1
```

5.29.1.30 CONFIG_AWS_IOT_SHADOW_MAX_JSON_TOKEN_EXPECTED

```
#define CONFIG_AWS_IOT_SHADOW_MAX_JSON_TOKEN_EXPECTED 120
```

5.29.1.31 CONFIG_AWS_IOT_SHADOW_MAX_SHADOW_TOPIC_LENGTH_WITHOUT_THINGNAME

```
#define CONFIG_AWS_IOT_SHADOW_MAX_SHADOW_TOPIC_LENGTH_WITHOUT_THINGNAME 60
```

5.29.1.32 CONFIG_AWS_IOT_SHADOW_MAX_SIMULTANEOUS_ACKS

```
#define CONFIG_AWS_IOT_SHADOW_MAX_SIMULTANEOUS_ACKS 10
```

5.29.1.33 CONFIG_AWS_IOT_SHADOW_MAX_SIMULTANEOUS_THINGNAMES

```
#define CONFIG_AWS_IOT_SHADOW_MAX_SIMULTANEOUS_THINGNAMES 10
```

5.29.1.34 CONFIG_AWS_IOT_SHADOW_MAX_SIZE_OF_THING_NAME

```
#define CONFIG_AWS_IOT_SHADOW_MAX_SIZE_OF_THING_NAME 20
```

5.29.1.35 CONFIG_AWS_IOT_SHADOW_MAX_SIZE_OF_UNIQUE_CLIENT_ID_BYTES

```
#define CONFIG_AWS_IOT_SHADOW_MAX_SIZE_OF_UNIQUE_CLIENT_ID_BYTES 80
```

5.29.1.36 CONFIG_BLE_ADV_REPORT_DISCARD_THRSHOLD

```
#define CONFIG_BLE_ADV_REPORT_DISCARD_THRSHOLD 20
```

5.29.1.37 CONFIG_BLE_ADV_REPORT_FLOW_CONTROL_NUM

```
#define CONFIG_BLE_ADV_REPORT_FLOW_CONTROL_NUM 100
```

5.29.1.38 CONFIG_BLE_ADV_REPORT_FLOW_CONTROL_SUPPORTED

```
#define CONFIG_BLE_ADV_REPORT_FLOW_CONTROL_SUPPORTED 1
```

5.29.1.39 CONFIG_BLE_ESTABLISH_LINK_CONNECTION_TIMEOUT

```
#define CONFIG_BLE_ESTABLISH_LINK_CONNECTION_TIMEOUT 30
```

5.29.1.40 CONFIG_BLE_SCAN_DUPLICATE

```
#define CONFIG_BLE_SCAN_DUPLICATE 1
```

5.29.1.41 CONFIG_BLE_SMP_ENABLE

```
#define CONFIG_BLE_SMP_ENABLE 1
```

5.29.1.42 CONFIG_BLUEDROID_ENABLED

```
#define CONFIG_BLUEDROID_ENABLED 1
```

5.29.1.43 CONFIG_BLUEDROID_PINNED_TO_CORE

```
#define CONFIG_BLUEDROID_PINNED_TO_CORE 0
```

5.29.1.44 CONFIG_BLUEDROID_PINNED_TO_CORE_0

```
#define CONFIG_BLUEDROID_PINNED_TO_CORE_0 1
```

5.29.1.45 CONFIG_BLUFI_INITIAL_TRACE_LEVEL

```
#define CONFIG_BLUFI_INITIAL_TRACE_LEVEL 2
```

5.29.1.46 CONFIG_BLUFI_TRACE_LEVEL_WARNING

```
#define CONFIG_BLUFI_TRACE_LEVEL_WARNING 1
```

5.29.1.47 CONFIG_BNEP_INITIAL_TRACE_LEVEL

```
#define CONFIG_BNEP_INITIAL_TRACE_LEVEL 2
```

5.29.1.48 CONFIG_BNEP_TRACE_LEVEL_WARNING

```
#define CONFIG_BNEP_TRACE_LEVEL_WARNING 1
```

5.29.1.49 CONFIG_BOOTLOADER_VDDSDIO_BOOST_1_9V

```
#define CONFIG_BOOTLOADER_VDDSDIO_BOOST_1_9V 1
```

5.29.1.50 CONFIG_BOOTLOADER_WDT_ENABLE

```
#define CONFIG_BOOTLOADER_WDT_ENABLE 1
```

5.29.1.51 CONFIG_BOOTLOADER_WDT_TIME_MS

```
#define CONFIG_BOOTLOADER_WDT_TIME_MS 9000
```

5.29.1.52 CONFIG_BROWNOUT_DET

```
#define CONFIG_BROWNOUT_DET 1
```

5.29.1.53 CONFIG_BROWNOUT_DET_LVL

```
#define CONFIG_BROWNOUT_DET_LVL 0
```

5.29.1.54 CONFIG_BROWNOUT_DET_LVL_SEL_0

```
#define CONFIG_BROWNOUT_DET_LVL_SEL_0 1
```

5.29.1.55 CONFIG_BT_ACL_CONNECTIONS

```
#define CONFIG_BT_ACL_CONNECTIONS 4
```

5.29.1.56 CONFIG_BT_ENABLED

```
#define CONFIG_BT_ENABLED 1
```

5.29.1.57 CONFIG_BT_RESERVE_DRAM

```
#define CONFIG_BT_RESERVE_DRAM 0xdb5c
```

5.29.1.58 CONFIG_BTC_INITIAL_TRACE_LEVEL

```
#define CONFIG_BTC_INITIAL_TRACE_LEVEL 2
```

5.29.1.59 CONFIG_BTC_TASK_STACK_SIZE

```
#define CONFIG_BTC_TASK_STACK_SIZE 3072
```

5.29.1.60 CONFIG_BTC_TRACE_LEVEL_WARNING

```
#define CONFIG_BTC_TRACE_LEVEL_WARNING 1
```

5.29.1.61 CONFIG_BTDM_CONTROLLER_BLE_MAX_CONN

```
#define CONFIG_BTDM_CONTROLLER_BLE_MAX_CONN 3
```

5.29.1.62 CONFIG_BTDM_CONTROLLER_BLE_MAX_CONN_EFF

```
#define CONFIG_BTDM_CONTROLLER_BLE_MAX_CONN_EFF 3
```

5.29.1.63 CONFIG_BTDM_CONTROLLER_BR_EDR_MAX_ACL_CONN_EFF

```
#define CONFIG_BTDM_CONTROLLER_BR_EDR_MAX_ACL_CONN_EFF 0
```

5.29.1.64 CONFIG_BTDM_CONTROLLER_BR_EDR_MAX_SYNC_CONN_EFF

```
#define CONFIG_BTDM_CONTROLLER_BR_EDR_MAX_SYNC_CONN_EFF 0
```

5.29.1.65 CONFIG_BTDM_CONTROLLER_HCI_MODE_VHCI

```
#define CONFIG_BTDM_CONTROLLER_HCI_MODE_VHCI 1
```

5.29.1.66 CONFIG_BTDM_CONTROLLER_MODE_BLE_ONLY

```
#define CONFIG_BTDM_CONTROLLER_MODE_BLE_ONLY 1
```

5.29.1.67 CONFIG_BTDM_CONTROLLER_PINNED_TO_CORE

```
#define CONFIG_BTDM_CONTROLLER_PINNED_TO_CORE 0
```

5.29.1.68 CONFIG_BTDM_CONTROLLER_PINNED_TO_CORE_0

```
#define CONFIG_BTDM_CONTROLLER_PINNED_TO_CORE_0 1
```

5.29.1.69 CONFIG_BTIF_INITIAL_TRACE_LEVEL

```
#define CONFIG_BTIF_INITIAL_TRACE_LEVEL 2
```

5.29.1.70 CONFIG_BTIF_TRACE_LEVEL_WARNING

```
#define CONFIG_BTIF_TRACE_LEVEL_WARNING 1
```

5.29.1.71 CONFIG_BTM_INITIAL_TRACE_LEVEL

```
#define CONFIG_BTM_INITIAL_TRACE_LEVEL 2
```

5.29.1.72 CONFIG_BTM_TRACE_LEVEL_WARNING

```
#define CONFIG_BTM_TRACE_LEVEL_WARNING 1
```

5.29.1.73 CONFIG_BTU_TASK_STACK_SIZE

```
#define CONFIG_BTU_TASK_STACK_SIZE 4096
```

5.29.1.74 CONFIG_CONSOLE_UART_BAUDRATE

```
#define CONFIG_CONSOLE_UART_BAUDRATE 115200
```

5.29.1.75 CONFIG_CONSOLE_UART_DEFAULT

```
#define CONFIG_CONSOLE_UART_DEFAULT 1
```

5.29.1.76 CONFIG_CONSOLE_UART_NUM

```
#define CONFIG_CONSOLE_UART_NUM 0
```

5.29.1.77 CONFIG_DMA_RX_BUF_NUM

```
#define CONFIG_DMA_RX_BUF_NUM 10
```

5.29.1.78 CONFIG_DMA_TX_BUF_NUM

```
#define CONFIG_DMA_TX_BUF_NUM 10
```

5.29.1.79 CONFIG_DUPLICATE_SCAN_CACHE_SIZE

```
#define CONFIG_DUPLICATE_SCAN_CACHE_SIZE 50
```

5.29.1.80 CONFIG_EFUSE_CODE_SCHEME_COMPAT_3_4

```
#define CONFIG_EFUSE_CODE_SCHEME_COMPAT_3_4 1
```

5.29.1.81 CONFIG_EFUSE_MAX_BLK_LEN

```
#define CONFIG_EFUSE_MAX_BLK_LEN 192
```

5.29.1.82 CONFIG_EMAC_CHECK_LINK_PERIOD_MS

```
#define CONFIG_EMAC_CHECK_LINK_PERIOD_MS 2000
```

5.29.1.83 CONFIG_EMAC_TASK_PRIORITY

```
#define CONFIG_EMAC_TASK_PRIORITY 20
```

5.29.1.84 CONFIG_EMAC_TASK_STACK_SIZE

```
#define CONFIG_EMAC_TASK_STACK_SIZE 3072
```

5.29.1.85 CONFIG_ENABLE_ARDUINO_DEPENDS

```
#define CONFIG_ENABLE_ARDUINO_DEPENDS 1
```

5.29.1.86 CONFIG_ESP32_APPTRACE_DEST_NONE

```
#define CONFIG_ESP32_APPTRACE_DEST_NONE 1
```

5.29.1.87 CONFIG_ESP32_APPTRACE_LOCK_ENABLE

```
#define CONFIG_ESP32_APPTRACE_LOCK_ENABLE 1
```

5.29.1.88 CONFIG_ESP32_DEBUG_OCDAWARE

```
#define CONFIG_ESP32_DEBUG_OCDAWARE 1
```

5.29.1.89 CONFIG_ESP32_DEBUG_STUBS_ENABLE

```
#define CONFIG_ESP32_DEBUG_STUBS_ENABLE 1
```

5.29.1.90 CONFIG_ESP32_DEEP_SLEEP_WAKEUP_DELAY

```
#define CONFIG_ESP32_DEEP_SLEEP_WAKEUP_DELAY 2000
```

5.29.1.91 CONFIG_ESP32_DEFAULT_CPU_FREQ_160

```
#define CONFIG_ESP32_DEFAULT_CPU_FREQ_160 1
```

5.29.1.92 CONFIG_ESP32_DEFAULT_CPU_FREQ_MHZ

```
#define CONFIG_ESP32_DEFAULT_CPU_FREQ_MHZ 160
```

5.29.1.93 CONFIG_ESP32_DEFAULT_PTHREAD_CORE_NO_AFFINITY

```
#define CONFIG_ESP32_DEFAULT_PTHREAD_CORE_NO_AFFINITY 1
```

5.29.1.94 CONFIG_ESP32_DPORT_WORKAROUND

```
#define CONFIG_ESP32_DPORT_WORKAROUND 1
```

5.29.1.95 CONFIG_ESP32_ENABLE_COREDUMP_TO_NONE

```
#define CONFIG_ESP32_ENABLE_COREDUMP_TO_NONE 1
```

5.29.1.96 CONFIG_ESP32_PANIC_PRINT_REBOOT

```
#define CONFIG_ESP32_PANIC_PRINT_REBOOT 1
```

5.29.1.97 CONFIG_ESP32_PHY_CALIBRATION_AND_DATA_STORAGE

```
#define CONFIG_ESP32_PHY_CALIBRATION_AND_DATA_STORAGE 1
```

5.29.1.98 CONFIG_ESP32_PHY_MAX_TX_POWER

```
#define CONFIG_ESP32_PHY_MAX_TX_POWER 20
```

5.29.1.99 CONFIG_ESP32_PHY_MAX_WIFI_TX_POWER

```
#define CONFIG_ESP32_PHY_MAX_WIFI_TX_POWER 20
```

5.29.1.100 CONFIG_ESP32_PTHREAD_TASK_CORE_DEFAULT

```
#define CONFIG_ESP32_PTHREAD_TASK_CORE_DEFAULT -1
```

5.29.1.101 CONFIG_ESP32_PTHREAD_TASK_NAME_DEFAULT

```
#define CONFIG_ESP32_PTHREAD_TASK_NAME_DEFAULT "pthread"
```

5.29.1.102 CONFIG_ESP32_PTHREAD_TASK_PRIO_DEFAULT

```
#define CONFIG_ESP32_PTHREAD_TASK_PRIO_DEFAULT 5
```

5.29.1.103 CONFIG_ESP32_PTHREAD_TASK_STACK_SIZE_DEFAULT

```
#define CONFIG_ESP32_PTHREAD_TASK_STACK_SIZE_DEFAULT 3072
```

5.29.1.104 CONFIG_ESP32_REV_MIN

```
#define CONFIG_ESP32_REV_MIN 0
```

5.29.1.105 CONFIG_ESP32_REV_MIN_0

```
#define CONFIG_ESP32_REV_MIN_0 1
```

5.29.1.106 CONFIG_ESP32_RTC_CLK_CAL_CYCLES

```
#define CONFIG_ESP32_RTC_CLK_CAL_CYCLES 1024
```

5.29.1.107 CONFIG_ESP32_RTC_CLOCK_SOURCE_INTERNAL_RC

```
#define CONFIG_ESP32_RTC_CLOCK_SOURCE_INTERNAL_RC 1
```

5.29.1.108 CONFIG_ESP32_TIME_SYSCALL_USE_RTC_FRC1

```
#define CONFIG_ESP32_TIME_SYSCALL_USE_RTC_FRC1 1
```

5.29.1.109 CONFIG_ESP32_WIFI_AMPDU_RX_ENABLED

```
#define CONFIG_ESP32_WIFI_AMPDU_RX_ENABLED 1
```

5.29.1.110 CONFIG_ESP32_WIFI_AMPDU_TX_ENABLED

```
#define CONFIG_ESP32_WIFI_AMPDU_TX_ENABLED 1
```

5.29.1.111 CONFIG_ESP32_WIFI_DYNAMIC_RX_BUFFER_NUM

```
#define CONFIG_ESP32_WIFI_DYNAMIC_RX_BUFFER_NUM 32
```

5.29.1.112 CONFIG_ESP32_WIFI_DYNAMIC_TX_BUFFER

```
#define CONFIG_ESP32_WIFI_DYNAMIC_TX_BUFFER 1
```

5.29.1.113 CONFIG_ESP32_WIFI_DYNAMIC_TX_BUFFER_NUM

```
#define CONFIG_ESP32_WIFI_DYNAMIC_TX_BUFFER_NUM 32
```

5.29.1.114 CONFIG_ESP32_WIFI_IRAM_OPT

```
#define CONFIG_ESP32_WIFI_IRAM_OPT 1
```

5.29.1.115 CONFIG_ESP32_WIFI_MGMT_SBUF_NUM

```
#define CONFIG_ESP32_WIFI_MGMT_SBUF_NUM 32
```

5.29.1.116 CONFIG_ESP32_WIFI_NVS_ENABLED

```
#define CONFIG_ESP32_WIFI_NVS_ENABLED 1
```

5.29.1.117 CONFIG_ESP32_WIFI_RX_BA_WIN

```
#define CONFIG_ESP32_WIFI_RX_BA_WIN 6
```

5.29.1.118 CONFIG_ESP32_WIFI_SOFTAP_BEACON_MAX_LEN

```
#define CONFIG_ESP32_WIFI_SOFTAP_BEACON_MAX_LEN 752
```

5.29.1.119 CONFIG_ESP32_WIFI_STATIC_RX_BUFFER_NUM

```
#define CONFIG_ESP32_WIFI_STATIC_RX_BUFFER_NUM 10
```

5.29.1.120 CONFIG_ESP32_WIFI_TASK_PINNED_TO_CORE_0

```
#define CONFIG_ESP32_WIFI_TASK_PINNED_TO_CORE_0 1
```

5.29.1.121 CONFIG_ESP32_WIFI_TX_BA_WIN

```
#define CONFIG_ESP32_WIFI_TX_BA_WIN 6
```

5.29.1.122 CONFIG_ESP32_WIFI_TX_BUFFER_TYPE

```
#define CONFIG_ESP32_WIFI_TX_BUFFER_TYPE 1
```

5.29.1.123 CONFIG_ESP32_XTAL_FREQ

```
#define CONFIG_ESP32_XTAL_FREQ 40
```

5.29.1.124 CONFIG_ESP32_XTAL_FREQ_40

```
#define CONFIG_ESP32_XTAL_FREQ_40 1
```

5.29.1.125 CONFIG_ESP_ERR_TO_NAME_LOOKUP

```
#define CONFIG_ESP_ERR_TO_NAME_LOOKUP 1
```

5.29.1.126 CONFIG_ESP_GRATUITOUS_ARP

```
#define CONFIG_ESP_GRATUITOUS_ARP 1
```

5.29.1.127 CONFIG_ESP_HTTP_CLIENT_ENABLE_HTTPS

```
#define CONFIG_ESP_HTTP_CLIENT_ENABLE_HTTPS 1
```

5.29.1.128 CONFIG_ESPTOOLPY_AFTER

```
#define CONFIG_ESPTOOLPY_AFTER "hard_reset"
```

5.29.1.129 CONFIG_ESPTOOLPY_AFTER_RESET

```
#define CONFIG_ESPTOOLPY_AFTER_RESET 1
```

5.29.1.130 CONFIG_ESPTOOLPY_BAUD

```
#define CONFIG_ESPTOOLPY_BAUD 115200
```

5.29.1.131 CONFIG_ESPTOOLPY_BAUD_115200B

```
#define CONFIG_ESPTOOLPY_BAUD_115200B 1
```

5.29.1.132 CONFIG_ESPTOOLPY_BAUD_OTHER_VAL

```
#define CONFIG_ESPTOOLPY_BAUD_OTHER_VAL 115200
```

5.29.1.133 CONFIG_ESPTOOLPY_BEFORE

```
#define CONFIG_ESPTOOLPY_BEFORE "default_reset"
```

5.29.1.134 CONFIG_ESPTOOLPY_BEFORE_RESET

```
#define CONFIG_ESPTOOLPY_BEFORE_RESET 1
```

5.29.1.135 CONFIG_ESPTOOLPY_COMPRESSED

```
#define CONFIG_ESPTOOLPY_COMPRESSED 1
```

5.29.1.136 CONFIG_ESPTOOLPY_FLASHFREQ

```
#define CONFIG_ESPTOOLPY_FLASHFREQ "40m"
```

5.29.1.137 CONFIG_ESPTOOLPY_FLASHFREQ_40M

```
#define CONFIG_ESPTOOLPY_FLASHFREQ_40M 1
```

5.29.1.138 CONFIG_ESPTOOLPY_FLASHMODE

```
#define CONFIG_ESPTOOLPY_FLASHMODE "dio"
```

5.29.1.139 CONFIG_ESPTOOLPY_FLASHSIZE

```
#define CONFIG_ESPTOOLPY_FLASHSIZE "2MB"
```

5.29.1.140 CONFIG_ESPTOOLPY_FLASHSIZE_2MB

```
#define CONFIG_ESPTOOLPY_FLASHSIZE_2MB 1
```

5.29.1.141 CONFIG_ESPTOOLPY_FLASHSIZE_DETECT

```
#define CONFIG_ESPTOOLPY_FLASHSIZE_DETECT 1
```

5.29.1.142 CONFIG_ESPTOOLPY_PORT

```
#define CONFIG_ESPTOOLPY_PORT "COM19"
```

5.29.1.143 CONFIG_FATFS_CODEPAGE

```
#define CONFIG_FATFS_CODEPAGE 437
```

5.29.1.144 CONFIG_FATFS_CODEPAGE_437

```
#define CONFIG_FATFS_CODEPAGE_437 1
```

5.29.1.145 CONFIG_FATFS_FS_LOCK

```
#define CONFIG_FATFS_FS_LOCK 0
```

5.29.1.146 CONFIG_FATFS_LFN_NONE

```
#define CONFIG_FATFS_LFN_NONE 1
```

5.29.1.147 CONFIG_FATFS_PER_FILE_CACHE

```
#define CONFIG_FATFS_PER_FILE_CACHE 1
```

5.29.1.148 CONFIG_FATFS_TIMEOUT_MS

```
#define CONFIG_FATFS_TIMEOUT_MS 10000
```

5.29.1.149 CONFIG_FLASHMODE_DIO

```
#define CONFIG_FLASHMODE_DIO 1
```

5.29.1.150 CONFIG_FOUR_UNIVERSAL_MAC_ADDRESS

```
#define CONFIG_FOUR_UNIVERSAL_MAC_ADDRESS 1
```

5.29.1.151 CONFIG_FREERTOS_ASSERT_FAIL_ABORT

```
#define CONFIG_FREERTOS_ASSERT_FAIL_ABORT 1
```

5.29.1.152 CONFIG_FREERTOS_ASSERT_ON_UNTESTED_FUNCTION

```
#define CONFIG_FREERTOS_ASSERT_ON_UNTESTED_FUNCTION 1
```

5.29.1.153 CONFIG_FREERTOS_CHECK_MUTEX_GIVEN_BY_OWNER

```
#define CONFIG_FREERTOS_CHECK_MUTEX_GIVEN_BY_OWNER 1
```

5.29.1.154 CONFIG_FREERTOS_CHECK_STACKOVERFLOW_CANARY

```
#define CONFIG_FREERTOS_CHECK_STACKOVERFLOW_CANARY 1
```

5.29.1.155 CONFIG_FREERTOS_CORETIMER_0

```
#define CONFIG_FREERTOS_CORETIMER_0 1
```

5.29.1.156 CONFIG_FREERTOS_HZ

```
#define CONFIG_FREERTOS_HZ 100
```

5.29.1.157 CONFIG_FREERTOS_IDLE_TASK_STACKSIZE

```
#define CONFIG_FREERTOS_IDLE_TASK_STACKSIZE 1536
```

5.29.1.158 CONFIG_FREERTOS_INTERRUPT_BACKTRACE

```
#define CONFIG_FREERTOS_INTERRUPT_BACKTRACE 1
```

5.29.1.159 CONFIG_FREERTOS_ISR_STACKSIZE

```
#define CONFIG_FREERTOS_ISR_STACKSIZE 1536
```

5.29.1.160 CONFIG_FREERTOS_MAX_TASK_NAME_LEN

```
#define CONFIG_FREERTOS_MAX_TASK_NAME_LEN 16
```

5.29.1.161 CONFIG_FREERTOS_NO_AFFINITY

```
#define CONFIG_FREERTOS_NO_AFFINITY 0x7FFFFFFF
```

5.29.1.162 CONFIG_FREERTOS_QUEUE_REGISTRY_SIZE

```
#define CONFIG_FREERTOS_QUEUE_REGISTRY_SIZE 0
```

5.29.1.163 CONFIG_FREERTOS_TASK_FUNCTION_WRAPPER

```
#define CONFIG_FREERTOS_TASK_FUNCTION_WRAPPER 1
```

5.29.1.164 CONFIG_FREERTOS_THREAD_LOCAL_STORAGE_POINTERS

```
#define CONFIG_FREERTOS_THREAD_LOCAL_STORAGE_POINTERS 1
```

5.29.1.165 CONFIG_GAP_INITIAL_TRACE_LEVEL

```
#define CONFIG_GAP_INITIAL_TRACE_LEVEL 2
```

5.29.1.166 CONFIG_GAP_TRACE_LEVEL_WARNING

```
#define CONFIG_GAP_TRACE_LEVEL_WARNING 1
```

5.29.1.167 CONFIG_GARP_TMR_INTERVAL

```
#define CONFIG_GARP_TMR_INTERVAL 60
```

5.29.1.168 CONFIG_GATT_INITIAL_TRACE_LEVEL

```
#define CONFIG_GATT_INITIAL_TRACE_LEVEL 2
```

5.29.1.169 CONFIG_GATT_TRACE_LEVEL_WARNING

```
#define CONFIG_GATT_TRACE_LEVEL_WARNING 1
```

5.29.1.170 CONFIG_GATTC_ENABLE

```
#define CONFIG_GATTC_ENABLE 1
```

5.29.1.171 CONFIG_GATTS_ENABLE

```
#define CONFIG_GATTS_ENABLE 1
```

5.29.1.172 CONFIG_GATTS_SEND_SERVICE_CHANGE_AUTO

```
#define CONFIG_GATTS_SEND_SERVICE_CHANGE_AUTO 1
```

5.29.1.173 CONFIG_GATTS_SEND_SERVICE_CHANGE_MODE

```
#define CONFIG_GATTS_SEND_SERVICE_CHANGE_MODE 0
```

5.29.1.174 CONFIG_HCI_INITIAL_TRACE_LEVEL

```
#define CONFIG_HCI_INITIAL_TRACE_LEVEL 2
```

5.29.1.175 CONFIG_HCI_TRACE_LEVEL_WARNING

```
#define CONFIG_HCI_TRACE_LEVEL_WARNING 1
```

5.29.1.176 CONFIG_HEAP_POISONING_DISABLED

```
#define CONFIG_HEAP_POISONING_DISABLED 1
```

5.29.1.177 CONFIG_HID_INITIAL_TRACE_LEVEL

```
#define CONFIG_HID_INITIAL_TRACE_LEVEL 2
```

5.29.1.178 CONFIG_HID_TRACE_LEVEL_WARNING

```
#define CONFIG_HID_TRACE_LEVEL_WARNING 1
```

5.29.1.179 CONFIG_HTTPD_ERR_RESP_NO_DELAY

```
#define CONFIG_HTTPD_ERR_RESP_NO_DELAY 1
```

5.29.1.180 CONFIG_HTTPD_MAX_REQ_HDR_LEN

```
#define CONFIG_HTTPD_MAX_REQ_HDR_LEN 512
```

5.29.1.181 CONFIG_HTTPD_MAX_URI_LEN

```
#define CONFIG_HTTPD_MAX_URI_LEN 512
```

5.29.1.182 CONFIG_HTTPD_PURGE_BUF_LEN

```
#define CONFIG_HTTPD_PURGE_BUF_LEN 32
```

5.29.1.183 CONFIG_IDF_TARGET

```
#define CONFIG_IDF_TARGET "esp32"
```

5.29.1.184 CONFIG_IDF_TARGET_ESP32

```
#define CONFIG_IDF_TARGET_ESP32 1
```

5.29.1.185 CONFIG_INT_WDT

```
#define CONFIG_INT_WDT 1
```

5.29.1.186 CONFIG_INT_WDT_CHECK_CPU1

```
#define CONFIG_INT_WDT_CHECK_CPU1 1
```

5.29.1.187 CONFIG_INT_WDT_TIMEOUT_MS

```
#define CONFIG_INT_WDT_TIMEOUT_MS 300
```

5.29.1.188 CONFIG_IP_LOST_TIMER_INTERVAL

```
#define CONFIG_IP_LOST_TIMER_INTERVAL 120
```

5.29.1.189 CONFIG_IPC_TASK_STACK_SIZE

```
#define CONFIG_IPC_TASK_STACK_SIZE 1024
```

5.29.1.190 CONFIG_L2CAP_INITIAL_TRACE_LEVEL

```
#define CONFIG_L2CAP_INITIAL_TRACE_LEVEL 2
```

5.29.1.191 CONFIG_L2CAP_TRACE_LEVEL_WARNING

```
#define CONFIG_L2CAP_TRACE_LEVEL_WARNING 1
```

5.29.1.192 CONFIG_LIBSODIUM_USE_MBEDTLS_SHA

```
#define CONFIG_LIBSODIUM_USE_MBEDTLS_SHA 1
```

5.29.1.193 CONFIG_LOG_BOOTLOADER_LEVEL

```
#define CONFIG_LOG_BOOTLOADER_LEVEL 3
```

5.29.1.194 CONFIG_LOG_BOOTLOADER_LEVEL_INFO

```
#define CONFIG_LOG_BOOTLOADER_LEVEL_INFO 1
```

5.29.1.195 CONFIG_LOG_COLORS

```
#define CONFIG_LOG_COLORS 1
```

5.29.1.196 CONFIG_LOG_DEFAULT_LEVEL

```
#define CONFIG_LOG_DEFAULT_LEVEL 3
```

5.29.1.197 CONFIG_LOG_DEFAULT_LEVEL_INFO

```
#define CONFIG_LOG_DEFAULT_LEVEL_INFO 1
```

5.29.1.198 CONFIG_LWIP_DHCP_DOES_ARP_CHECK

```
#define CONFIG_LWIP_DHCP_DOES_ARP_CHECK 1
```

5.29.1.199 CONFIG_LWIP_DHCP_MAX_NTP_SERVERS

```
#define CONFIG_LWIP_DHCP_MAX_NTP_SERVERS 1
```

5.29.1.200 CONFIG_LWIP_DHCPS_LEASE_UNIT

```
#define CONFIG_LWIP_DHCPS_LEASE_UNIT 60
```

5.29.1.201 CONFIG_LWIP_DHCPS_MAX_STATION_NUM

```
#define CONFIG_LWIP_DHCPS_MAX_STATION_NUM 8
```

5.29.1.202 CONFIG_LWIP_LOOPBACK_MAX_PBUFS

```
#define CONFIG_LWIP_LOOPBACK_MAX_PBUFS 8
```

5.29.1.203 CONFIG_LWIP_MAX_ACTIVE_TCP

```
#define CONFIG_LWIP_MAX_ACTIVE_TCP 16
```

5.29.1.204 CONFIG_LWIP_MAX_LISTENING_TCP

```
#define CONFIG_LWIP_MAX_LISTENING_TCP 16
```

5.29.1.205 CONFIG_LWIP_MAX_RAW_PCBS

```
#define CONFIG_LWIP_MAX_RAW_PCBS 16
```

5.29.1.206 CONFIG_LWIP_MAX_SOCKETS

```
#define CONFIG_LWIP_MAX_SOCKETS 10
```

5.29.1.207 CONFIG_LWIP_MAX_UDP_PCBS

```
#define CONFIG_LWIP_MAX_UDP_PCBS 16
```

5.29.1.208 CONFIG_LWIP_NETIF_LOOPBACK

```
#define CONFIG_LWIP_NETIF_LOOPBACK 1
```

5.29.1.209 CONFIG_LWIP_SO_REUSE

```
#define CONFIG_LWIP_SO_REUSE 1
```

5.29.1.210 CONFIG_LWIP_SO_REUSE_RXTOALL

```
#define CONFIG_LWIP_SO_REUSE_RXTOALL 1
```

5.29.1.211 CONFIG_MAIN_TASK_STACK_SIZE

```
#define CONFIG_MAIN_TASK_STACK_SIZE 3584
```

5.29.1.212 CONFIG_MAKE_WARN_UNDEFINED_VARIABLES

```
#define CONFIG_MAKE_WARN_UNDEFINED_VARIABLES 1
```

5.29.1.213 CONFIG_MB_CONTROLLER_NOTIFY_QUEUE_SIZE

```
#define CONFIG_MB_CONTROLLER_NOTIFY_QUEUE_SIZE 20
```

5.29.1.214 CONFIG_MB_CONTROLLER_NOTIFY_TIMEOUT

```
#define CONFIG_MB_CONTROLLER_NOTIFY_TIMEOUT 20
```

5.29.1.215 CONFIG_MB_CONTROLLER_STACK_SIZE

```
#define CONFIG_MB_CONTROLLER_STACK_SIZE 4096
```

5.29.1.216 CONFIG_MB_EVENT_QUEUE_TIMEOUT

```
#define CONFIG_MB_EVENT_QUEUE_TIMEOUT 20
```

5.29.1.217 CONFIG_MB_QUEUE_LENGTH

```
#define CONFIG_MB_QUEUE_LENGTH 20
```

5.29.1.218 CONFIG_MB_SERIAL_BUF_SIZE

```
#define CONFIG_MB_SERIAL_BUF_SIZE 256
```

5.29.1.219 CONFIG_MB_SERIAL_TASK_PRIO

```
#define CONFIG_MB_SERIAL_TASK_PRIO 10
```

5.29.1.220 CONFIG_MB_SERIAL_TASK_STACK_SIZE

```
#define CONFIG_MB_SERIAL_TASK_STACK_SIZE 2048
```

5.29.1.221 CONFIG_MB_TIMER_GROUP

```
#define CONFIG_MB_TIMER_GROUP 0
```

5.29.1.222 CONFIG_MB_TIMER_INDEX

```
#define CONFIG_MB_TIMER_INDEX 0
```

5.29.1.223 CONFIG_MB_TIMER_PORT_ENABLED

```
#define CONFIG_MB_TIMER_PORT_ENABLED 1
```

5.29.1.224 CONFIG_MBEDTLS_AES_C

```
#define CONFIG_MBEDTLS_AES_C 1
```

5.29.1.225 CONFIG_MBEDTLS_CCM_C

```
#define CONFIG_MBEDTLS_CCM_C 1
```

5.29.1.226 CONFIG_MBEDTLS_ECDH_C

```
#define CONFIG_MBEDTLS_ECDH_C 1
```

5.29.1.227 CONFIG_MBEDTLS_ECDSA_C

```
#define CONFIG_MBEDTLS_ECDSA_C 1
```

5.29.1.228 CONFIG_MBEDTLS_ECP_C

```
#define CONFIG_MBEDTLS_ECP_C 1
```

5.29.1.229 CONFIG_MBEDTLS_ECP_DP_BP256R1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_BP256R1_ENABLED 1
```

5.29.1.230 CONFIG_MBEDTLS_ECP_DP_BP384R1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_BP384R1_ENABLED 1
```

5.29.1.231 CONFIG_MBEDTLS_ECP_DP_BP512R1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_BP512R1_ENABLED 1
```

5.29.1.232 CONFIG_MBEDTLS_ECP_DP_CURVE25519_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_CURVE25519_ENABLED 1
```

5.29.1.233 CONFIG_MBEDTLS_ECP_DP_SECP192K1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_SECP192K1_ENABLED 1
```

5.29.1.234 CONFIG_MBEDTLS_ECP_DP_SECP192R1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_SECP192R1_ENABLED 1
```

5.29.1.235 CONFIG_MBEDTLS_ECP_DP_SECP224K1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_SECP224K1_ENABLED 1
```

5.29.1.236 CONFIG_MBEDTLS_ECP_DP_SECP224R1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_SECP224R1_ENABLED 1
```

5.29.1.237 CONFIG_MBEDTLS_ECP_DP_SECP256K1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_SECP256K1_ENABLED 1
```

5.29.1.238 CONFIG_MBEDTLS_ECP_DP_SECP256R1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_SECP256R1_ENABLED 1
```

5.29.1.239 CONFIG_MBEDTLS_ECP_DP_SECP384R1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_SECP384R1_ENABLED 1
```

5.29.1.240 CONFIG_MBEDTLS_ECP_DP_SECP521R1_ENABLED

```
#define CONFIG_MBEDTLS_ECP_DP_SECP521R1_ENABLED 1
```

5.29.1.241 CONFIG_MBEDTLS_ECP_NIST_OPTIM

```
#define CONFIG_MBEDTLS_ECP_NIST_OPTIM 1
```

5.29.1.242 CONFIG_MBEDTLS_GCM_C

```
#define CONFIG_MBEDTLS_GCM_C 1
```

5.29.1.243 CONFIG_MBEDTLS_HARDWARE_AES

```
#define CONFIG_MBEDTLS_HARDWARE_AES 1
```

5.29.1.244 CONFIG_MBEDTLS_HAVE_TIME

```
#define CONFIG_MBEDTLS_HAVE_TIME 1
```

5.29.1.245 CONFIG_MBEDTLS_INTERNAL_MEM_ALLOC

```
#define CONFIG_MBEDTLS_INTERNAL_MEM_ALLOC 1
```

5.29.1.246 CONFIG_MBEDTLS_KEY_EXCHANGE_DHE_RSA

```
#define CONFIG_MBEDTLS_KEY_EXCHANGE_DHE_RSA 1
```

5.29.1.247 CONFIG_MBEDTLS_KEY_EXCHANGE_ECDH_ECDSA

```
#define CONFIG_MBEDTLS_KEY_EXCHANGE_ECDH_ECDSA 1
```

5.29.1.248 CONFIG_MBEDTLS_KEY_EXCHANGE_ECDH_RSA

```
#define CONFIG_MBEDTLS_KEY_EXCHANGE_ECDH_RSA 1
```

5.29.1.249 CONFIG_MBEDTLS_KEY_EXCHANGE_ECDHE_ECDSA

```
#define CONFIG_MBEDTLS_KEY_EXCHANGE_ECDHE_ECDSA 1
```

5.29.1.250 CONFIG_MBEDTLS_KEY_EXCHANGE_ECDHE_RSA

```
#define CONFIG_MBEDTLS_KEY_EXCHANGE_ECDHE_RSA 1
```

5.29.1.251 CONFIG_MBEDTLS_KEY_EXCHANGE_ELLIPTIC_CURVE

```
#define CONFIG_MBEDTLS_KEY_EXCHANGE_ELLIPTIC_CURVE 1
```

5.29.1.252 CONFIG_MBEDTLS_KEY_EXCHANGE_RSA

```
#define CONFIG_MBEDTLS_KEY_EXCHANGE_RSA 1
```

5.29.1.253 CONFIG_MBEDTLS_PEM_PARSE_C

```
#define CONFIG_MBEDTLS_PEM_PARSE_C 1
```

5.29.1.254 CONFIG_MBEDTLS_PEM_WRITE_C

```
#define CONFIG_MBEDTLS_PEM_WRITE_C 1
```

5.29.1.255 CONFIG_MBEDTLS_RC4_DISABLED

```
#define CONFIG_MBEDTLS_RC4_DISABLED 1
```

5.29.1.256 CONFIG_MBEDTLS_SSL_ALPN

```
#define CONFIG_MBEDTLS_SSL_ALPN 1
```

5.29.1.257 CONFIG_MBEDTLS_SSL_MAX_CONTENT_LEN

```
#define CONFIG_MBEDTLS_SSL_MAX_CONTENT_LEN 16384
```

5.29.1.258 CONFIG_MBEDTLS_SSL_PROTO_TLS1

```
#define CONFIG_MBEDTLS_SSL_PROTO_TLS1 1
```

5.29.1.259 CONFIG_MBEDTLS_SSL_PROTO_TLS1_1

```
#define CONFIG_MBEDTLS_SSL_PROTO_TLS1_1 1
```

5.29.1.260 CONFIG_MBEDTLS_SSL_PROTO_TLS1_2

```
#define CONFIG_MBEDTLS_SSL_PROTO_TLS1_2 1
```

5.29.1.261 CONFIG_MBEDTLS_SSL_RENEGOTIATION

```
#define CONFIG_MBEDTLS_SSL_RENEGOTIATION 1
```

5.29.1.262 CONFIG_MBEDTLS_SSL_SESSION_TICKETS

```
#define CONFIG_MBEDTLS_SSL_SESSION_TICKETS 1
```

5.29.1.263 CONFIG_MBEDTLS_TLS_CLIENT

```
#define CONFIG_MBEDTLS_TLS_CLIENT 1
```

5.29.1.264 CONFIG_MBEDTLS_TLS_ENABLED

```
#define CONFIG_MBEDTLS_TLS_ENABLED 1
```

5.29.1.265 CONFIG_MBEDTLS_TLS_SERVER

```
#define CONFIG_MBEDTLS_TLS_SERVER 1
```

5.29.1.266 CONFIG_MBEDTLS_TLS_SERVER_AND_CLIENT

```
#define CONFIG_MBEDTLS_TLS_SERVER_AND_CLIENT 1
```

5.29.1.267 CONFIG_MBEDTLS_X509_CRL_PARSE_C

```
#define CONFIG_MBEDTLS_X509_CRL_PARSE_C 1
```

5.29.1.268 CONFIG_MBEDTLS_X509_CSR_PARSE_C

```
#define CONFIG_MBEDTLS_X509_CSR_PARSE_C 1
```

5.29.1.269 CONFIG_MCA_INITIAL_TRACE_LEVEL

```
#define CONFIG_MCA_INITIAL_TRACE_LEVEL 2
```

5.29.1.270 CONFIG_MCA_TRACE_LEVEL_WARNING

```
#define CONFIG_MCA_TRACE_LEVEL_WARNING 1
```

5.29.1.271 CONFIG_MDNS_MAX_SERVICES

```
#define CONFIG_MDNS_MAX_SERVICES 10
```

5.29.1.272 CONFIG_MONITOR_BAUD

```
#define CONFIG_MONITOR_BAUD 115200
```

5.29.1.273 CONFIG_MONITOR_BAUD_115200B

```
#define CONFIG_MONITOR_BAUD_115200B 1
```

5.29.1.274 CONFIG_MONITOR_BAUD_OTHER_VAL

```
#define CONFIG_MONITOR_BAUD_OTHER_VAL 115200
```

5.29.1.275 CONFIG_MQTT_PROTOCOL_311

```
#define CONFIG_MQTT_PROTOCOL_311 1
```

5.29.1.276 CONFIG_MQTT_TRANSPORT_SSL

```
#define CONFIG_MQTT_TRANSPORT_SSL 1
```

5.29.1.277 CONFIG_MQTT_TRANSPORT_WEBSOCKET

```
#define CONFIG_MQTT_TRANSPORT_WEBSOCKET 1
```

5.29.1.278 CONFIG_MQTT_TRANSPORT_WEBSOCKET_SECURE

```
#define CONFIG_MQTT_TRANSPORT_WEBSOCKET_SECURE 1
```

5.29.1.279 CONFIG_NEWLIB_STDIN_LINE_ENDING_CR

```
#define CONFIG_NEWLIB_STDIN_LINE_ENDING_CR 1
```

5.29.1.280 CONFIG_NEWLIB_STDOUT_LINE_ENDING_CRLF

```
#define CONFIG_NEWLIB_STDOUT_LINE_ENDING_CRLF 1
```

5.29.1.281 CONFIG_NUMBER_OF_UNIVERSAL_MAC_ADDRESS

```
#define CONFIG_NUMBER_OF_UNIVERSAL_MAC_ADDRESS 4
```

5.29.1.282 CONFIG_OPENSSL_ASSERT_DO_NOTHING

```
#define CONFIG_OPENSSL_ASSERT_DO_NOTHING 1
```

5.29.1.283 CONFIG_OPTIMIZATION_ASSERTIONS_ENABLED

```
#define CONFIG_OPTIMIZATION_ASSERTIONS_ENABLED 1
```

5.29.1.284 CONFIG_OPTIMIZATION_LEVEL_DEBUG

```
#define CONFIG_OPTIMIZATION_LEVEL_DEBUG 1
```

5.29.1.285 CONFIG_OSI_INITIAL_TRACE_LEVEL

```
#define CONFIG_OSI_INITIAL_TRACE_LEVEL 2
```

5.29.1.286 CONFIG_OSI_TRACE_LEVEL_WARNING

```
#define CONFIG_OSI_TRACE_LEVEL_WARNING 1
```

5.29.1.287 CONFIG_PAN_INITIAL_TRACE_LEVEL

```
#define CONFIG_PAN_INITIAL_TRACE_LEVEL 2
```

5.29.1.288 CONFIG_PAN_TRACE_LEVEL_WARNING

```
#define CONFIG_PAN_TRACE_LEVEL_WARNING 1
```

5.29.1.289 CONFIG_PARTITION_TABLE_CUSTOM_FILENAME

```
#define CONFIG_PARTITION_TABLE_CUSTOM_FILENAME "partitions.csv"
```

5.29.1.290 CONFIG_PARTITION_TABLE_FILENAME

```
#define CONFIG_PARTITION_TABLE_FILENAME "partitions_singleapp.csv"
```

5.29.1.291 CONFIG_PARTITION_TABLE_MD5

```
#define CONFIG_PARTITION_TABLE_MD5 1
```

5.29.1.292 CONFIG_PARTITION_TABLE_OFFSET

```
#define CONFIG_PARTITION_TABLE_OFFSET 0x8000
```

5.29.1.293 CONFIG_PARTITION_TABLE_SINGLE_APP

```
#define CONFIG_PARTITION_TABLE_SINGLE_APP 1
```

5.29.1.294 CONFIG_PTHREAD_STACK_MIN

```
#define CONFIG_PTHREAD_STACK_MIN 768
```

5.29.1.295 CONFIG_PYTHON

```
#define CONFIG_PYTHON "python"
```

5.29.1.296 CONFIG_REDUCE_PHY_TX_POWER

```
#define CONFIG_REDUCE_PHY_TX_POWER 1
```

5.29.1.297 CONFIG_RFCOMM_INITIAL_TRACE_LEVEL

```
#define CONFIG_RFCOMM_INITIAL_TRACE_LEVEL 2
```

5.29.1.298 CONFIG_RFCOMM_TRACE_LEVEL_WARNING

```
#define CONFIG_RFCOMM_TRACE_LEVEL_WARNING 1
```

5.29.1.299 CONFIG_SCAN_DUPLICATE_BY_DEVICE_ADDR

```
#define CONFIG_SCAN_DUPLICATE_BY_DEVICE_ADDR 1
```

5.29.1.300 CONFIG_SCAN_DUPLICATE_TYPE

```
#define CONFIG_SCAN_DUPLICATE_TYPE 0
```

5.29.1.301 CONFIG_SDP_INITIAL_TRACE_LEVEL

```
#define CONFIG_SDP_INITIAL_TRACE_LEVEL 2
```

5.29.1.302 CONFIG_SDP_TRACE_LEVEL_WARNING

```
#define CONFIG_SDP_TRACE_LEVEL_WARNING 1
```

5.29.1.303 CONFIG_SMP_ENABLE

```
#define CONFIG_SMP_ENABLE 1
```

5.29.1.304 CONFIG_SMP_INITIAL_TRACE_LEVEL

```
#define CONFIG_SMP_INITIAL_TRACE_LEVEL 2
```

5.29.1.305 CONFIG_SMP_TRACE_LEVEL_WARNING

```
#define CONFIG_SMP_TRACE_LEVEL_WARNING 1
```

5.29.1.306 CONFIG_SPI_FLASH_ERASE_YIELD_DURATION_MS

```
#define CONFIG_SPI_FLASH_ERASE_YIELD_DURATION_MS 20
```

5.29.1.307 CONFIG_SPI_FLASH_ERASE_YIELD_TICKS

```
#define CONFIG_SPI_FLASH_ERASE_YIELD_TICKS 1
```

5.29.1.308 CONFIG_SPI_FLASH_ROM_DRIVER_PATCH

```
#define CONFIG_SPI_FLASH_ROM_DRIVER_PATCH 1
```

5.29.1.309 CONFIG_SPI_FLASH_WRITING_DANGEROUS_REGIONS_ABORTS

```
#define CONFIG_SPI_FLASH_WRITING_DANGEROUS_REGIONS_ABORTS 1
```

5.29.1.310 CONFIG_SPI_FLASH_YIELD_DURING_ERASE

```
#define CONFIG_SPI_FLASH_YIELD_DURING_ERASE 1
```

5.29.1.311 CONFIG_SPI_MASTER_ISR_IN_IRAM

```
#define CONFIG_SPI_MASTER_ISR_IN_IRAM 1
```

5.29.1.312 CONFIG_SPI_SLAVE_ISR_IN_IRAM

```
#define CONFIG_SPI_SLAVE_ISR_IN_IRAM 1
```

5.29.1.313 CONFIG_SPIFFS_CACHE

```
#define CONFIG_SPIFFS_CACHE 1
```

5.29.1.314 CONFIG_SPIFFS_CACHE_WR

```
#define CONFIG_SPIFFS_CACHE_WR 1
```

5.29.1.315 CONFIG_SPIFFS_GC_MAX_RUNS

```
#define CONFIG_SPIFFS_GC_MAX_RUNS 10
```

5.29.1.316 CONFIG_SPIFFS_MAX_PARTITIONS

```
#define CONFIG_SPIFFS_MAX_PARTITIONS 3
```

5.29.1.317 CONFIG_SPIFFS_META_LENGTH

```
#define CONFIG_SPIFFS_META_LENGTH 4
```

5.29.1.318 CONFIG_SPIFFS_OBJ_NAME_LEN

```
#define CONFIG_SPIFFS_OBJ_NAME_LEN 32
```

5.29.1.319 CONFIG_SPIFFS_PAGE_CHECK

```
#define CONFIG_SPIFFS_PAGE_CHECK 1
```

5.29.1.320 CONFIG_SPIFFS_PAGE_SIZE

```
#define CONFIG_SPIFFS_PAGE_SIZE 256
```

5.29.1.321 CONFIG_SPIFFS_USE_MAGIC

```
#define CONFIG_SPIFFS_USE_MAGIC 1
```

5.29.1.322 CONFIG_SPIFFS_USE_MAGIC_LENGTH

```
#define CONFIG_SPIFFS_USE_MAGIC_LENGTH 1
```

5.29.1.323 CONFIG_SPIFFS_USE_MTIME

```
#define CONFIG_SPIFFS_USE_MTIME 1
```

5.29.1.324 CONFIG_STACK_CHECK_NONE

```
#define CONFIG_STACK_CHECK_NONE 1
```

5.29.1.325 CONFIG_SUPPORT_TERMIOS

```
#define CONFIG_SUPPORT_TERMIOS 1
```

5.29.1.326 CONFIG_SUPPRESS_SELECT_DEBUG_OUTPUT

```
#define CONFIG_SUPPRESS_SELECT_DEBUG_OUTPUT 1
```

5.29.1.327 CONFIG_SW_COEXIST_ENABLE

```
#define CONFIG_SW_COEXIST_ENABLE 1
```

5.29.1.328 CONFIG_SW_COEXIST_PREFERENCE_BALANCE

```
#define CONFIG_SW_COEXIST_PREFERENCE_BALANCE 1
```

5.29.1.329 CONFIG_SW_COEXIST_PREFERENCE_VALUE

```
#define CONFIG_SW_COEXIST_PREFERENCE_VALUE 2
```

5.29.1.330 CONFIG_SYSTEM_EVENT_QUEUE_SIZE

```
#define CONFIG_SYSTEM_EVENT_QUEUE_SIZE 32
```

5.29.1.331 CONFIG_SYSTEM_EVENT_TASK_STACK_SIZE

```
#define CONFIG_SYSTEM_EVENT_TASK_STACK_SIZE 2304
```

5.29.1.332 CONFIG_TASK_WDT

```
#define CONFIG_TASK_WDT 1
```

5.29.1.333 CONFIG_TASK_WDT_CHECK_IDLE_TASK_CPU0

```
#define CONFIG_TASK_WDT_CHECK_IDLE_TASK_CPU0 1
```

5.29.1.334 CONFIG_TASK_WDT_CHECK_IDLE_TASK_CPU1

```
#define CONFIG_TASK_WDT_CHECK_IDLE_TASK_CPU1 1
```

5.29.1.335 CONFIG_TASK_WDT_TIMEOUT_S

```
#define CONFIG_TASK_WDT_TIMEOUT_S 5
```

5.29.1.336 CONFIG_TCP_MAXRTX

```
#define CONFIG_TCP_MAXRTX 12
```

5.29.1.337 CONFIG_TCP_MSL

```
#define CONFIG_TCP_MSL 60000
```

5.29.1.338 CONFIG_TCP_MSS

```
#define CONFIG_TCP_MSS 1436
```

5.29.1.339 CONFIG_TCP_OVERSIZE_MSS

```
#define CONFIG_TCP_OVERSIZE_MSS 1
```

5.29.1.340 CONFIG_TCP_QUEUE_OOSEQ

```
#define CONFIG_TCP_QUEUE_OOSEQ 1
```

5.29.1.341 CONFIG_TCP_RECVMBOX_SIZE

```
#define CONFIG_TCP_RECVMBOX_SIZE 6
```

5.29.1.342 CONFIG_TCP_SND_BUF_DEFAULT

```
#define CONFIG_TCP_SND_BUF_DEFAULT 5744
```

5.29.1.343 CONFIG_TCP_SYNMAXRTX

```
#define CONFIG_TCP_SYNMAXRTX 6
```

5.29.1.344 CONFIG_TCP_WND_DEFAULT

```
#define CONFIG_TCP_WND_DEFAULT 5744
```

5.29.1.345 CONFIG_TCPIP_LWIP

```
#define CONFIG_TCPIP_LWIP 1
```

5.29.1.346 CONFIG_TCPIP_RECVMBOX_SIZE

```
#define CONFIG_TCPIP_RECVMBOX_SIZE 32
```

5.29.1.347 CONFIG_TCPIP_TASK_AFFINITY

```
#define CONFIG_TCPIP_TASK_AFFINITY 0x7FFFFFFF
```

5.29.1.348 CONFIG_TCPIP_TASK_AFFINITY_NO_AFFINITY

```
#define CONFIG_TCPIP_TASK_AFFINITY_NO_AFFINITY 1
```

5.29.1.349 CONFIG_TCPIP_TASK_STACK_SIZE

```
#define CONFIG_TCPIP_TASK_STACK_SIZE 2048
```

5.29.1.350 CONFIG_TIMER_QUEUE_LENGTH

```
#define CONFIG_TIMER_QUEUE_LENGTH 10
```

5.29.1.351 CONFIG_TIMER_TASK_PRIORITY

```
#define CONFIG_TIMER_TASK_PRIORITY 1
```

5.29.1.352 CONFIG_TIMER_TASK_STACK_DEPTH

```
#define CONFIG_TIMER_TASK_STACK_DEPTH 2048
```

5.29.1.353 CONFIG_TIMER_TASK_STACK_SIZE

```
#define CONFIG_TIMER_TASK_STACK_SIZE 3584
```

5.29.1.354 CONFIG_TOOLPREFIX

```
#define CONFIG_TOOLPREFIX "xtensa-esp32-elf-"
```

5.29.1.355 CONFIG_TRACEMEM_RESERVE_DRAM

```
#define CONFIG_TRACEMEM_RESERVE_DRAM 0x0
```

5.29.1.356 CONFIG_UDP_RECVMBOX_SIZE

```
#define CONFIG_UDP_RECVMBOX_SIZE 6
```

5.29.1.357 CONFIG_ULP_COPROC_RESERVE_MEM

```
#define CONFIG_ULP_COPROC_RESERVE_MEM 0
```

5.29.1.358 CONFIG_UNITY_ENABLE_DOUBLE

```
#define CONFIG_UNITY_ENABLE_DOUBLE 1
```

5.29.1.359 CONFIG_UNITY_ENABLE_FLOAT

```
#define CONFIG_UNITY_ENABLE_FLOAT 1
```

5.29.1.360 CONFIG_UNITY_ENABLE_IDF_TEST_RUNNER

```
#define CONFIG_UNITY_ENABLE_IDF_TEST_RUNNER 1
```

5.29.1.361 CONFIG_WIFI_PROV_SCAN_MAX_ENTRIES

```
#define CONFIG_WIFI_PROV_SCAN_MAX_ENTRIES 16
```

5.29.1.362 CONFIG_WL_SECTOR_SIZE

```
#define CONFIG_WL_SECTOR_SIZE 4096
```

5.29.1.363 CONFIG_WL_SECTOR_SIZE_4096

```
#define CONFIG_WL_SECTOR_SIZE_4096 1
```

5.30 src/wifi.cpp File Reference

File to connect ESP32 to WiFi.

```
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event_loop.h"
#include "esp_log.h"
#include "wifi.h"
```

Macros

- #define ESP_WIFI_SSID "Esp32"
- #define ESP_WIFI_PASS "hello123"
- #define MAX_STA_CONN 4

Functions

- void wifi_init_softap ()
- void wifi_init_sta ()

Variables

- const int WIFI_CONNECTED_BIT = BIT0

5.30.1 Detailed Description

File to connect ESP32 to WiFi.

Author

Andy Yang (andyyeung123@gmail.com)

Version

0.1

Date

2020-04-27

Copyright

Copyright (c) 2020

5.30.2 Macro Definition Documentation

5.30.2.1 ESP_WIFI_PASS

```
#define ESP_WIFI_PASS "hello123"
```

5.30.2.2 ESP_WIFI_SSID

```
#define ESP_WIFI_SSID "Esp32"
```

5.30.2.3 MAX_STA_CONN

```
#define MAX_STA_CONN 4
```

5.30.3 Function Documentation

5.30.3.1 wifi_init_softap()

```
void wifi_init_softap ( )
```

5.30.3.2 wifi_init_sta()

```
void wifi_init_sta ( )
```

5.30.4 Variable Documentation

5.30.4.1 WIFI_CONNECTED_BIT

```
const int WIFI_CONNECTED_BIT = BIT0
```

Index