network_msg_struct

Generated by Doxygen 1.9.6

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 cfg_state_t Union Reference

a BLE Mesh configuration state with parameters defined according to Mesh Specifications

`#include <network_msg_struct.h>`

Public Attributes

- esp_ble_mesh_cfg_client_get_state_t **get_state**
- esp_ble_mesh_cfg_client_set_state_t **set_state**

3.1.1 Detailed Description

a BLE Mesh configuration state with parameters defined according to Mesh Specifications

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

- network_msg_struct.h

Chapter 4

File Documentation

4.1 my_crc.h File Reference

CRC-16/CCITT algorithm.

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

Functions

- uint16_t CRCCCITT (uint8_t *data, size_t length_of_interest)

  Runs a CRC-16-CCITT checksum algorithm. It will return and generate 2 bytes of checksum if you are sender. If you are receiver, run the function with the length_of_interest accounting for the last 2 bytes of crc. It should return 0 to indicate no data loss for 99.9984% of the time.

4.1.1 Detailed Description

CRC-16/CCITT algorithm.

Author

Shane

Version

0.1

Date

2023-02-23

Copyright

4.1.2 Function Documentation

4.1.2.1 CRCCCITT()

```
uint16_t CRCCCITT (
            uint8_t * data,
            size_t length_of_interest )
```

Runs a CRC-16-CCITT checksum algorithm. It will return and generate 2 bytes of checksum if you are sender. If you are receiver, run the function with the length_of_interest accounting for the last 2 bytes of crc. It should return 0 to indicate no data loss for 99.9984% of the time.

Parameters

data	uffer with a size of the message contents + 2 bytes for CRC portion. This buffer's data must be in little endian format regardless of checking or receiving
length_of_interest	number of bytes of interest in message content(exclude CRC part) if generating CRC. If checking CRC, must account for the number of bytes for CRC in length

Returns

uint16_t For sender: returns crc checksum value. For receiver: returns 0 if no data loss

4.2 my_crc.h

Go to the documentation of this file.

```
00001
00012 #ifndef MY_CRC_H
00013 #define MY_CRC_H
00014
00015 #include <stddef.h>
00016 #include <stdint.h>
00017
00018  // Name   : "CRC-16/CITT"
00019  //  Width  : 16
00020  //  Poly   : 1021
00021  //  Init   : FFFF
00022  //  RefIn  : False
00023  //  RefOut : False
00024  //  XorOut : 0000
00025
00035 uint16_t CRCCCITT(uint8_t *data, size_t length_of_interest);
00036
00037 #endif
```

4.3 network_msg_struct.h File Reference

helper functions to pack/unpack BLE Mesh messages(format A, format C)

```
#include "esp_ble_mesh_defs.h"
#include "esp_ble_mesh_config_model_api.h"
#include "my_custom_models_def.h"
```

Classes

- union cfg_state_t

  a BLE Mesh configuration state with parameters defined according to Mesh Specifications

Functions

- uint32_t extract_opcode (uint8_t *buf)

  extract opcode segment from any message.
- uint16_t extract_addr (uint8_t *buf)

  extract address segment from message A.
- int32_t extract_sensor_data_msgA (uint8_t *buf, model_sensor_data_t *sensor_buf)

  extract sensor data payload from buf to store in sensor_buf
- int32_t extract_bt_data_msgA (uint8_t *buf, cfg_state_t *state)

  extract bluetooth config data payload from buf to store in cfg_state_t state
- uint8_t * set_sensor_data_msgA (uint32_t opcode, uint16_t addr, model_sensor_data_t *sensor_buf)

  Set the buffer containing sensor message A , which includes generating and setting the crc in message A.
- uint8_t * set_bt_data_msgA (uint32_t opcode, uint16_t addr, cfg_state_t *state)

  Set the buffer containing bt message A , which includes generating and setting the crc in message A.
- int32_t extract_sensor_data_msgC (uint8_t *buf, model_sensor_data_t *sensor_buf)

  extract sensor data payload from buf to store in sensor_buf
- uint8_t * set_sensor_data_msgC (uint32_t opcode, model_sensor_data_t *sensor_buf)

  Set the buffer containing message C , which includes generating and setting the crc in message C.

4.3.1 Detailed Description

helper functions to pack/unpack BLE Mesh messages(format A, format C)

Author

Shane

Version

0.1

Date

2023-02-22

Copyright

4.3.2 Function Documentation

4.3.2.1 extract_addr()

```
uint16_t extract_addr (
            uint8_t * buf )
```

extract address segment from message A.

Parameters

buf	pointer to message A

Returns

uint16_t unicast addr or group addr, 0 if fail

4.3.2.2 extract_bt_data_msgA()

```
int32_t extract_bt_data_msgA (
            uint8_t * buf,
            cfg_state_t * state )
```

extract bluetooth config data payload from buf to store in cfg_state_t state

Parameters

buf	buf should have enough data allocated and user should ensure it contains message A bluetooth config msg format
state	empty buffer to store bt config data, must be initialised first

Returns

returns 0 if successful, -1 if fail. Note: returning 0 does not guarantee succcess unless buf contains correct data

4.3.2.3 extract_opcode()

```
uint32_t extract_opcode (
            uint8_t * buf )
```

extract opcode segment from any message.

Parameters

buf	pointer to message

Returns

uint32_t opcode, 0 if fail

4.3.2.4 extract_sensor_data_msgA()

```
int32_t extract_sensor_data_msgA (
            uint8_t * buf,
            model_sensor_data_t * sensor_buf )
```

extract sensor data payload from buf to store in sensor_buf

Parameters

buf	buf should have enough data allocated and user should ensure it contains message A sensor msg format
sensor_buf	empty buffer to store sensor data, must be initialised first

Returns

returns 0 if successful, -1 if fail. Note: returning 0 does not guarantee succcess unless buf contains correct data

4.3.2.5 extract_sensor_data_msgC()

```
int32_t extract_sensor_data_msgC (
            uint8_t * buf,
            model_sensor_data_t * sensor_buf )
```

extract sensor data payload from buf to store in sensor_buf

Parameters

buf	buf should have enough data allocated and user should ensure it contains message C msg format
sensor_buf	empty buffer to store sensor data, must be initialised first

Returns

returns 0 if successful, -1 if fail. Note: returning 0 does not guarantee succcess unless buf contains correct data

4.3.2.6 set_bt_data_msgA()

```
uint8_t * set_bt_data_msgA (
            uint32_t opcode,
            uint16_t addr,
            cfg_state_t * state )
```

Set the buffer containing bt message A , which includes generating and setting the crc in message A.

Parameters

opcode	opcode of command
addr	destination address
state	valid bt config data pointer

Returns

returns pointer to buffer(static define, not thread-safe) containing message A if successful, NULL if fail. Note: returning valid pointer does not guarantee succcess unless state contains correct data

4.3.2.7 set_sensor_data_msgA()

```
uint8_t * set_sensor_data_msgA (
            uint32_t opcode,
            uint16_t addr,
            model_sensor_data_t * sensor_buf )
```

Set the buffer containing sensor message A , which includes generating and setting the crc in message A.

Parameters

opcode	opcode of command
addr	destination address
sensor_buf	valid sensor data pointer

Returns

returns pointer to buffer(static define, not thread-safe) containing message A if successful, NULL if fail. Note: returning valid pointer does not guarantee succcess unless sensor_buf contains correct data

4.3.2.8 set_sensor_data_msgC()

```
uint8_t * set_sensor_data_msgC (
            uint32_t opcode,
            model_sensor_data_t * sensor_buf )
```

Set the buffer containing message C , which includes generating and setting the crc in message C.

Parameters

opcode	opcode of command
sensor_buf	valid sensor data pointer

Returns

returns pointer to buffer(static define, not thread-safe) containing message C if successful, NULL if fail. Note: returning valid pointer does not guarantee succcess unless sensor_buf contains correct data

4.4 network_msg_struct.h

Go to the documentation of this file.

```
00001
00012 #pragma once
00013 #ifndef NET_MSG_STRUCT
00014 #define NET_MSG_STRUCT
00015
00016 #if defined (__GLIBC__)
00017 # include <endian.h>
00018 # if (__BYTE_ORDER == __LITTLE_ENDIAN)
00019 #else
00020 #error "invalid endianness"
00021 #endif
00022 #endif
00023
00024 #include "esp_ble_mesh_defs.h"
00025 #include "esp_ble_mesh_config_model_api.h"
00026 #include "my_custom_models_def.h"
00027
00028
00029 #ifdef __cplusplus
00030 extern "C" {
00031 #endif
00032
00037 typedef union
00038 {
00039   esp_ble_mesh_cfg_client_get_state_t get_state;
00040   esp_ble_mesh_cfg_client_set_state_t set_state;
00041 }cfg_state_t;
00042
00049 uint32_t extract_opcode(uint8_t *buf);
00050
00057 uint16_t extract_addr(uint8_t *buf);
00058
00066 int32_t extract_sensor_data_msgA(uint8_t *buf, model_sensor_data_t *sensor_buf);
00067
00075 int32_t extract_bt_data_msgA(uint8_t* buf, cfg_state_t* state);
00076
00085 uint8_t* set_sensor_data_msgA(uint32_t opcode, uint16_t addr, model_sensor_data_t *sensor_buf);
00086
00095 uint8_t* set_bt_data_msgA(uint32_t opcode, uint16_t addr, cfg_state_t *state);
00096
00104 int32_t extract_sensor_data_msgC(uint8_t *buf, model_sensor_data_t* sensor_buf);
00105
00113 uint8_t* set_sensor_data_msgC(uint32_t opcode, model_sensor_data_t *sensor_buf);
00114
00115
00116 #ifdef __cplusplus
00117 }
00118 #endif
00119
00120 #endif /* NET_MSG_STRUCT */
```

Index