

CanSat Software

0.1

Generated on Tue Jan 13 2026 for CanSat Software by Doxygen 1.16.0

Tue Jan 13 2026 11:13:13

1 Directory Hierarchy	1
1.1 Directories	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Directory Documentation	7
4.1 code Directory Reference	7
5 Class Documentation	9
5.1 current_time_t Struct Reference	9
5.1.1 Member Data Documentation	9
5.1.1.1 day	9
5.1.1.2 hour	9
5.1.1.3 min	9
5.1.1.4 month	10
5.1.1.5 photo_count	10
5.1.1.6 sec	10
5.1.1.7 year	10
5.2 gps_data_t Struct Reference	10
5.2.1 Detailed Description	11
5.2.2 Member Data Documentation	11
5.2.2.1 altitude	11
5.2.2.2 fix	11
5.2.2.3 hour	11
5.2.2.4 latitude	11
5.2.2.5 longitude	11
5.2.2.6 min	11
5.2.2.7 satellites	12
5.2.2.8 sec	12
5.3 sensor_readings_t Struct Reference	12
5.3.1 Member Data Documentation	12
5.3.1.1 altitude_m	12
5.3.1.2 ammonia_ppm	12
5.3.1.3 humidity_pct	12
5.3.1.4 methane_ppm	13
5.3.1.5 oxygen_pct	13
5.3.1.6 pressure_pa	13
5.3.1.7 temperature_c	13
5.3.1.8 timestamp_str	13

6 File Documentation	15
6.1 code/atm_sen_module.c File Reference	15
6.1.1 Macro Definition Documentation	15
6.1.1.1 I2C_PORT	15
6.1.1.2 O2_ADDR	16
6.1.1.3 PIN_AMMONIA	16
6.1.1.4 PIN_METHANE	16
6.1.1.5 PIN_SCL	16
6.1.1.6 PIN_SDA	16
6.1.1.7 SHTC3_ADDR	16
6.1.2 Function Documentation	16
6.1.2.1 init_all_sensors()	16
6.1.2.2 read_all()	16
6.2 code/atm_sen_module.h File Reference	16
6.2.1 Function Documentation	17
6.2.1.1 init_all_sensors()	17
6.2.1.2 read_all()	17
6.3 atm_sen_module.h	17
6.4 code/debug_mode.h File Reference	17
6.4.1 Macro Definition Documentation	18
6.4.1.1 DEBUG_MODE	18
6.4.1.2 LOG	18
6.5 debug_mode.h	18
6.6 code/gps_module.c File Reference	18
6.6.1 Macro Definition Documentation	19
6.6.1.1 NMEA_BUFFER_LEN	19
6.6.2 Function Documentation	19
6.6.2.1 gps_get_data()	19
6.6.2.2 gps_init()	19
6.6.2.3 gps_update()	19
6.7 code/gps_module.h File Reference	20
6.7.1 Detailed Description	20
6.7.2 Macro Definition Documentation	20
6.7.2.1 GPS_BAUD_RATE	20
6.7.2.2 GPS_RX_PIN	21
6.7.2.3 GPS_TX_PIN	21
6.7.2.4 GPS_UART_ID	21
6.7.3 Function Documentation	21
6.7.3.1 gps_get_data()	21
6.7.3.2 gps_init()	21
6.7.3.3 gps_update()	22
6.8 gps_module.h	22

6.9 code/hw_config.c File Reference	22
6.9.1 Function Documentation	23
6.9.1.1 sd_get_by_num()	23
6.9.1.2 sd_get_num()	23
6.9.1.3 spi_get_by_num()	23
6.9.1.4 spi_get_num()	23
6.10 code/main.c File Reference	23
6.10.1 Function Documentation	24
6.10.1.1 main()	24
6.11 code/microsd_module.c File Reference	24
6.11.1 Function Documentation	24
6.11.1.1 save_gps_log()	24
6.11.1.2 save_system_data()	24
6.11.1.3 sd_init()	24
6.11.2 Variable Documentation	25
6.11.2.1 fs	25
6.11.2.2 is_mounted	25
6.12 code/microsd_module.h File Reference	25
6.12.1 Function Documentation	25
6.12.1.1 save_gps_log()	25
6.12.1.2 save_system_data()	25
6.12.1.3 sd_init()	25
6.13 microsd_module.h	26
6.14 code/time_manager.c File Reference	26
6.14.1 Macro Definition Documentation	26
6.14.1.1 TIMEZONE_OFFSET	26
6.14.2 Function Documentation	26
6.14.2.1 time_manager_get()	26
6.14.2.2 time_manager_init()	27
6.14.2.3 time_manager_sync()	27
6.14.2.4 time_manager_update()	27
6.15 code/time_manager.h File Reference	27
6.15.1 Function Documentation	27
6.15.1.1 time_manager_get()	27
6.15.1.2 time_manager_init()	28
6.15.1.3 time_manager_sync()	28
6.15.1.4 time_manager_update()	28
6.16 time_manager.h	28

Chapter 1

Directory Hierarchy

1.1 Directories

code	7
atm_sen_module.c	15
atm_sen_module.h	16
debug_mode.h	17
gps_module.c	18
gps_module.h	20
hw_config.c	22
main.c	23
microsd_module.c	24
microsd_module.h	25
time_manager.c	26
time_manager.h	27

Chapter 2

Class Index

2.1 Class List

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

current_time_t	9
gps_data_t	
These structures keep the parsed GPS position and time information	10
sensor_readings_t	12

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

code/atm_sen_module.c	15
code/atm_sen_module.h	16
code/debug_mode.h	17
code/gps_module.c	18
code/gps_module.h	
GPS module interface	20
code/hw_config.c	22
code/main.c	23
code/microsd_module.c	24
code/microsd_module.h	25
code/time_manager.c	26
code/time_manager.h	27

Chapter 4

Directory Documentation

4.1 code Directory Reference

Files

- file [atm_sen_module.c](#)
- file [atm_sen_module.h](#)
- file [debug_mode.h](#)
- file [gps_module.c](#)
- file [gps_module.h](#)
 - GPS module interface.*
- file [hw_config.c](#)
- file [main.c](#)
- file [microsd_module.c](#)
- file [microsd_module.h](#)
- file [time_manager.c](#)
- file [time_manager.h](#)

Chapter 5

Class Documentation

5.1 `current_time_t` Struct Reference

```
#include <time_manager.h>
```

Public Attributes

- `uint16_t` [year](#)
- `uint8_t` [month](#)
- `uint8_t` [day](#)
- `uint8_t` [hour](#)
- `uint8_t` [min](#)
- `uint8_t` [sec](#)
- `uint16_t` [photo_count](#)

5.1.1 Member Data Documentation

5.1.1.1 `day`

```
uint8_t current_time_t::day
```

5.1.1.2 `hour`

```
uint8_t current_time_t::hour
```

5.1.1.3 `min`

```
uint8_t current_time_t::min
```

5.1.1.4 month

```
uint8_t current_time_t::month
```

5.1.1.5 photo_count

```
uint16_t current_time_t::photo_count
```

5.1.1.6 sec

```
uint8_t current_time_t::sec
```

5.1.1.7 year

```
uint16_t current_time_t::year
```

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

- code/[time_manager.h](#)

5.2 gps_data_t Struct Reference

These structures keep the parsed GPS position and time information.

```
#include <gps_module.h>
```

Public Attributes

- float [latitude](#)
- float [longitude](#)
Latitude in decimal degrees.
- float [altitude](#)
Longitude in decimal degrees.
- uint8_t [satellites](#)
Altitude above mean sea level in meters [m].
- uint8_t [hour](#)
Number of satellites currently in view.
- uint8_t [min](#)
UTC Hour.
- uint8_t [sec](#)
UTC Minute.
- bool [fix](#)
UTC Second.

5.2.1 Detailed Description

These structures keep the parsed GPS position and time information.

The structure is updated whenever a valid NMEA sentence (\$GPRMC in this case) is successfully parsed.

5.2.2 Member Data Documentation

5.2.2.1 altitude

```
float gps_data_t::altitude
```

Longitude in decimal degrees.

5.2.2.2 fix

```
bool gps_data_t::fix
```

UTC Second.

5.2.2.3 hour

```
uint8_t gps_data_t::hour
```

Number of satellites currently in view.

5.2.2.4 latitude

```
float gps_data_t::latitude
```

5.2.2.5 longitude

```
float gps_data_t::longitude
```

Latitude in decimal degrees.

5.2.2.6 min

```
uint8_t gps_data_t::min
```

UTC Hour.

5.2.2.7 satellites

```
uint8_t gps_data_t::satellites
```

Altitude above mean sea level in meters [m].

5.2.2.8 sec

```
uint8_t gps_data_t::sec
```

UTC Minute.

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

- [code/gps_module.h](#)

5.3 sensor_readings_t Struct Reference

```
#include <atm_sen_module.h>
```

Public Attributes

- char [timestamp_str](#) [20]
- double [pressure_pa](#)
- double [altitude_m](#)
- float [temperature_c](#)
- float [humidity_pct](#)
- float [methane_ppm](#)
- float [ammonia_ppm](#)
- float [oxygen_pct](#)

5.3.1 Member Data Documentation

5.3.1.1 altitude_m

```
double sensor_readings_t::altitude_m
```

5.3.1.2 ammonia_ppm

```
float sensor_readings_t::ammonia_ppm
```

5.3.1.3 humidity_pct

```
float sensor_readings_t::humidity_pct
```

5.3.1.4 methane_ppm

```
float sensor_readings_t::methane_ppm
```

5.3.1.5 oxygen_pct

```
float sensor_readings_t::oxygen_pct
```

5.3.1.6 pressure_pa

```
double sensor_readings_t::pressure_pa
```

5.3.1.7 temperature_c

```
float sensor_readings_t::temperature_c
```

5.3.1.8 timestamp_str

```
char sensor_readings_t::timestamp_str[20]
```

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

- [code/atm_sen_module.h](#)

Chapter 6

File Documentation

6.1 code/atm_sen_module.c File Reference

```
#include <math.h>
#include "hardware/i2c.h"
#include "hardware/adc.h"
#include "atm_sen_module.h"
#include "debug_mode.h"
#include "libs/bmp280/bmp280_i2c.h"
#include "libs/bmp280/bmp280_i2c_hal.h"
#include "dfrobot_oxygen.h"
```

Macros

- `#define I2C_PORT i2c0`
- `#define SHTC3_ADDR 0x70`
- `#define PIN_SDA 12`
- `#define PIN_SCL 13`
- `#define O2_ADDR 0x74`
- `#define PIN_METHANE 26`
- `#define PIN_AMMONIA 27`

Functions

- void `init_all_sensors` ()
- void `read_all` (`sensor_readings_t` *gathered_data)

6.1.1 Macro Definition Documentation

6.1.1.1 I2C_PORT

```
#define I2C_PORT i2c0
```

6.1.1.2 O2_ADDR

```
#define O2_ADDR 0x74
```

6.1.1.3 PIN_AMMONIA

```
#define PIN_AMMONIA 27
```

6.1.1.4 PIN_METHANE

```
#define PIN_METHANE 26
```

6.1.1.5 PIN_SCL

```
#define PIN_SCL 13
```

6.1.1.6 PIN_SDA

```
#define PIN_SDA 12
```

6.1.1.7 SHTC3_ADDR

```
#define SHTC3_ADDR 0x70
```

6.1.2 Function Documentation

6.1.2.1 init_all_sensors()

```
void init_all_sensors (  
    void )
```

6.1.2.2 read_all()

```
void read_all (  
    sensor_readings_t * gathered_data)
```

6.2 code/atm_sen_module.h File Reference

```
#include <stdio.h>  
#include <pico/stdlib.h>  
#include <stdint.h>
```

Classes

- struct [sensor_readings_t](#)

Functions

- void [init_all_sensors](#) (void)
- void [read_all](#) ([sensor_readings_t](#) *gathered_data)

6.2.1 Function Documentation

6.2.1.1 init_all_sensors()

```
void init_all_sensors (
    void ) [extern]
```

6.2.1.2 read_all()

```
void read_all (
    sensor\_readings\_t * gathered_data) [extern]
```

6.3 atm_sen_module.h

[Go to the documentation of this file.](#)

```
00001 #ifndef ATM_SEN_MODULE_H
00002 #define ATM_SEN_MODULE_H
00003
00004 #include <stdio.h>
00005 #include <pico/stdlib.h>
00006 #include <stdint.h>
00007
00008 typedef struct
00009 {
00010     char timestamp_str[20]; // Time since boot in milliseconds
00011     double pressure_pa;
00012     double altitude_m;
00013     float temperature_c;
00014     float humidity_pct;
00015     float methane_ppm;
00016     float ammonia_ppm;
00017     float oxygen_pct;
00018 } sensor\_readings\_t;
00019
00020 extern void init\_all\_sensors(void);
00021 extern void read\_all(sensor\_readings\_t *gathered_data);
00022
00023 #endif
```

6.4 code/debug_mode.h File Reference

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

Macros

- `#define` [DEBUG_MODE](#)
- `#define` [LOG\(...\)](#)

6.4.1 Macro Definition Documentation

6.4.1.1 `DEBUG_MODE`

```
#define DEBUG_MODE
```

6.4.1.2 `LOG`

```
#define LOG(  
    ...)
```

Value:

```
printf(__VA_ARGS__)
```

6.5 `debug_mode.h`

[Go to the documentation of this file.](#)

```
00001 #ifndef DEBUG_MODE_H
00002 #define DEBUG_MODE_H
00003
00004 #include <stdio.h>
00005
00006 #define DEBUG_MODE
00007
00008 #ifdef DEBUG_MODE
00009     #define LOG(...) printf(__VA_ARGS__)
00010 #else
00011     #define LOG(...) ((void)0)
00012 #endif
00013
00014 #endif
```

6.6 `code/gps_module.c` File Reference

```
#include <string.h>
#include "gps_module.h"
#include "time_manager.h"
#include "debug_mode.h"
#include "minmea.h"
#include "hardware/uart.h"
```

Macros

- `#define` [NMEA_BUFFER_LEN](#) 85

Functions

- void `gps_init` (void)
Initializes the GPS UART connection and GPIO pins.
- bool `gps_update` (void)
Polls the UART for newly acquired GPS data and sends it to be parsed.
- void `gps_get_data` (`gps_data_t` *data)
Retrieves the latest parsed GPS data.

6.6.1 Macro Definition Documentation

6.6.1.1 NMEA_BUFFER_LEN

```
#define NMEA_BUFFER_LEN 85
```

6.6.2 Function Documentation

6.6.2.1 `gps_get_data()`

```
void gps_get_data (
    gps_data_t * data)
```

Retrieves the latest parsed GPS data.

Copies the most recent valid GPS data into the provided `gps_data_t` structure.

Parameters

out	data	Pointer to a <code>gps_data_t</code> struct where the data will be copied.
-----	------	--

6.6.2.2 `gps_init()`

```
void gps_init (
    void )
```

Initializes the GPS UART connection and GPIO pins.

Sets up the specified `GPS_UART_ID` with the baud rate defined in `GPS_BAUD_RATE` and configures the TX/RX pins.

6.6.2.3 `gps_update()`

```
bool gps_update (
    void )
```

Polls the UART for newly acquired GPS data and sends it to be parsed.

This function should be called frequently (e.g., in the main loop). It reads available characters from the UART buffer and calls a function to process NMEA sentences.

Returns

- true if a valid packet was fully parsed and data was updated.
- false if no new complete packet is available yet.

6.7 code/gps_module.h File Reference

GPS module interface.

```
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include "pico/stdlib.h"
```

Classes

- struct [gps_data_t](#)

These structures keep the parsed GPS position and time information.

Macros

- #define [GPS_BAUD_RATE](#) 9600
UART Baud rate for the GPS module (Standard is usually 9600).
- #define [GPS_UART_ID](#) uart0
The hardware UART instance to use (uart0 or uart1).
- #define [GPS_TX_PIN](#) 8
GPIO pin for UART transmission (Pico TX).
- #define [GPS_RX_PIN](#) 9
GPIO pin for UART reception (Pico RX).

Functions

- void [gps_init](#) (void)
Initializes the GPS UART connection and GPIO pins.
- bool [gps_update](#) (void)
Polls the UART for newly aquired GPS data and sends it to be parsed.
- void [gps_get_data](#) ([gps_data_t](#) *data)
Retrieves the latest parsed GPS data.

6.7.1 Detailed Description

GPS module interface.

This file handles the UART configuration for the GPS module (Pico), defines the data structure for holding parsed GPS coordinates/time, and provides functions to initialize and update the system.

6.7.2 Macro Definition Documentation

6.7.2.1 GPS_BAUD_RATE

```
#define GPS_BAUD_RATE 9600
```

UART Baud rate for the GPS module (Standard is usually 9600).

6.7.2.2 GPS_RX_PIN

```
#define GPS_RX_PIN 9
```

GPIO pin for UART reception (Pico RX).

6.7.2.3 GPS_TX_PIN

```
#define GPS_TX_PIN 8
```

GPIO pin for UART transmission (Pico TX).

6.7.2.4 GPS_UART_ID

```
#define GPS_UART_ID uart0
```

The hardware UART instance to use (uart0 or uart1).

6.7.3 Function Documentation

6.7.3.1 gps_get_data()

```
void gps_get_data (
    gps_data_t * data) [extern]
```

Retrieves the latest parsed GPS data.

Copies the most recent valid GPS data into the provided `gps_data_t` structure.

Parameters

out	data	Pointer to a gps_data_t struct where the data will be copied.
-----	------	---

6.7.3.2 gps_init()

```
void gps_init (
    void ) [extern]
```

Initializes the GPS UART connection and GPIO pins.

Sets up the specified `GPS_UART_ID` with the baud rate defined in `GPS_BAUD_RATE` and configures the TX/RX pins.

6.7.3.3 gps_update()

```
bool gps_update (
    void ) [extern]
```

Polls the UART for newly acquired GPS data and sends it to be parsed.

This function should be called frequently (e.g., in the main loop). It reads available characters from the UART buffer and calls a function to process NMEA sentences.

Returns

- true if a valid packet was fully parsed and data was updated.
- false if no new complete packet is available yet.

6.8 gps_module.h

[Go to the documentation of this file.](#)

```
00001
00010
00011 #ifndef GPS_MODULE_H
00012 #define GPS_MODULE_H
00013
00014 #include <stdint.h>
00015 #include <stdbool.h>
00016 #include <stdio.h>
00017 #include "pico/stdlib.h"
00018
00019 // CONFIGURATION MACROS
00020
00022 #define GPS_BAUD_RATE 9600
00023
00025 #define GPS_UART_ID uart0
00026
00028 #define GPS_TX_PIN 8
00029
00031 #define GPS_RX_PIN 9
00032
00033 // DATA STRUCTURES
00034
00040 typedef struct
00041 {
00042     float latitude;
00043     float longitude;
00044     float altitude;
00045     uint8_t satellites;
00046     uint8_t hour;
00047     uint8_t min;
00048     uint8_t sec;
00049     bool fix;
00050 } gps_data_t;
00051
00052 // FUNCTIONS
00053
00060
00061 extern void gps_init(void);
00062
00073 extern bool gps_update(void);
00074
00082 extern void gps_get_data(gps_data_t* data);
00083
00084 #endif // GPS_MODULE_H
```

6.9 code/hw_config.c File Reference

```
#include <assert.h>
#include <string.h>
#include "my_debug.h"
#include "hw_config.h"
#include "ff.h"
#include "diskio.h"
```

Functions

- `size_t sd_get_num ()`
- `sd_card_t * sd_get_by_num (size_t num)`
- `size_t spi_get_num ()`
- `spi_t * spi_get_by_num (size_t num)`

6.9.1 Function Documentation

6.9.1.1 sd_get_by_num()

```
sd_card_t * sd_get_by_num (  
    size_t num)
```

6.9.1.2 sd_get_num()

```
size_t sd_get_num ()
```

6.9.1.3 spi_get_by_num()

```
spi_t * spi_get_by_num (  
    size_t num)
```

6.9.1.4 spi_get_num()

```
size_t spi_get_num ()
```

6.10 code/main.c File Reference

```
#include <stdio.h>  
#include "pico/stdlib.h"  
#include "debug_mode.h"  
#include "atm_sen_module.h"  
#include "microsd_module.h"  
#include "time_manager.h"  
#include "gps_module.h"
```

Functions

- `int main (void)`

6.10.1 Function Documentation

6.10.1.1 main()

```
int main (
    void )
```

6.11 code/microsd_module.c File Reference

```
#include <string.h>
#include "microsd_module.h"
#include "debug_mode.h"
#include "ff.h"
#include "f_util.h"
#include "diskio.h"
#include "hw_config.h"
#include "hardware/spi.h"
```

Functions

- bool [sd_init](#) ()
- void [save_system_data](#) ([sensor_readings_t](#) *data, [current_time_t](#) *time)
- void [save_gps_log](#) ([gps_data_t](#) *gps)

6.11.1 Function Documentation

6.11.1.1 save_gps_log()

```
void save_gps_log (
    gps\_data\_t * gps)
```

6.11.1.2 save_system_data()

```
void save_system_data (
    sensor\_readings\_t * data,
    current\_time\_t * time)
```

6.11.1.3 sd_init()

```
bool sd_init ()
```

6.11.2 Variable Documentation

6.11.2.1 fs

FATFS fs

6.11.2.2 is_mounted

bool is_mounted

6.12 code/microsd_module.h File Reference

```
#include <stdbool.h>
#include <stdio.h>
#include <stdint.h>
#include "pico/stdlib.h"
#include "atm_sen_module.h"
#include "time_manager.h"
#include "gps_module.h"
```

Functions

- bool [sd_init](#) ()
- void [save_system_data](#) ([sensor_readings_t](#) *sen_data, [current_time_t](#) *time_date)
- void [save_gps_log](#) ([gps_data_t](#) *gps)

6.12.1 Function Documentation

6.12.1.1 save_gps_log()

```
void save_gps_log (
    gps\_data\_t * gps) [extern]
```

6.12.1.2 save_system_data()

```
void save_system_data (
    sensor\_readings\_t * sen_data,
    current\_time\_t * time_date) [extern]
```

6.12.1.3 sd_init()

```
bool sd_init () [extern]
```

6.13 microsd_module.h

[Go to the documentation of this file.](#)

```

00001 #ifndef MICROSD_MODULE_H
00002 #define MICROSD_MODULE_H
00003
00004 #include <stdbool.h>
00005 #include <stdio.h>
00006 #include <stdint.h>
00007 #include "pico/stdlib.h"
00008 #include "atm_sen_module.h"
00009 #include "time_manager.h"
00010 #include "gps_module.h"
00011
00012 extern bool sd_init();
00013
00014 extern void save_system_data(sensor_readings_t* sen_data, current_time_t* time_date);
00015
00016 //extern void save_photo(current_time_t* state, uint8_t* image_buffer, uint32_t length);
00017
00018 extern void save_gps_log(gps_data_t* gps);
00019
00020 #endif

```

6.14 code/time_manager.c File Reference

```

#include "time_manager.h"
#include "pico/stdlib.h"
#include <stdio.h>

```

Macros

- #define [TIMEZONE_OFFSET](#) 1

Functions

- void [time_manager_init](#) (void)
- bool [time_manager_update](#) (void)
- [current_time_t](#) * [time_manager_get](#) (void)
- void [time_manager_sync](#) (uint8_t hour, uint8_t min, uint8_t sec)

6.14.1 Macro Definition Documentation

6.14.1.1 TIMEZONE_OFFSET

```
#define TIMEZONE_OFFSET 1
```

6.14.2 Function Documentation

6.14.2.1 time_manager_get()

```

current\_time\_t * time\_manager\_get (
    void )

```


6.14.2.2 time_manager_init()

```
void time_manager_init (  
    void )
```

6.14.2.3 time_manager_sync()

```
void time_manager_sync (  
    uint8_t hour,  
    uint8_t min,  
    uint8_t sec)
```

6.14.2.4 time_manager_update()

```
bool time_manager_update (  
    void )
```

6.15 code/time_manager.h File Reference

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

Classes

- struct [current_time_t](#)

Functions

- void [time_manager_init](#) (void)
- bool [time_manager_update](#) (void)
- void [time_manager_sync](#) (uint8_t hour, uint8_t min, uint8_t sec)
- [current_time_t](#) * [time_manager_get](#) (void)

6.15.1 Function Documentation

6.15.1.1 time_manager_get()

```
current\_time\_t * time_manager_get (  
    void ) [extern]
```

6.15.1.2 time_manager_init()

```
void time_manager_init (  
    void ) [extern]
```

6.15.1.3 time_manager_sync()

```
void time_manager_sync (  
    uint8_t hour,  
    uint8_t min,  
    uint8_t sec) [extern]
```

6.15.1.4 time_manager_update()

```
bool time_manager_update (  
    void ) [extern]
```

6.16 time_manager.h

[Go to the documentation of this file.](#)

```
00001 #ifndef TIME_MANAGER_H  
00002 #define TIME_MANAGER_H  
00003  
00004 #include <stdint.h>  
00005 #include <stdbool.h>  
00006 #include <time.h>  
00007  
00008 typedef struct  
00009 {  
00010     uint16_t year;  
00011     uint8_t month;  
00012     uint8_t day;  
00013     uint8_t hour;  
00014     uint8_t min;  
00015     uint8_t sec;  
00016     uint16_t photo_count;  
00017 } current_time_t;  
00018  
00019 extern void time_manager_init(void);  
00020  
00021 extern bool time_manager_update(void);  
00022  
00023 extern void time_manager_sync(uint8_t hour, uint8_t min, uint8_t sec);  
00024  
00025 extern current_time_t* time_manager_get(void);  
00026  
00027 #endif
```