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

Chapter 1

Directory Hierarchy

1.1 Directories

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

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 senstence ($GPRMC in this case) is succesfully 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.

```
#ifndef ATM_SEN_MODULE_H
#define ATM_SEN_MODULE_H

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

typedef struct
{
    char timestamp_str[20]; // Time since boot in milliseconds
    double pressure_pa;
    double altitude_m;
    float temperature_c;
    float humidity_pct;
    float methane_ppm;
    float ammonia_ppm;
    float oxygen_pct;
} sensor_readings_t;

extern void init_all_sensors(void);
extern void read_all(sensor_readings_t *gathered_data);

#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.

```
#ifndef DEBUG_MODE_H
#define DEBUG_MODE_H

#include <stdio.h>

#define DEBUG_MODE

#ifdef DEBUG_MODE
    #define LOG(...) printf(__VA_ARGS__)
#else
    #define LOG(...) ((void)0)
#endif

#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 aquired 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 gos_data_t structure.

Parameters

out	*data*	Pointer to a gps_data_t 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 aquired 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 gos_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 aquired 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.

```

#ifndef GPS_MODULE_H
#define GPS_MODULE_H

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

// CONFIGURATION MACROS

#define GPS_BAUD_RATE 9600

#define GPS_UART_ID uart0

#define GPS_TX_PIN 8

#define GPS_RX_PIN 9

// DATA STRUCTURES

typedef struct
{
    float latitude;
    float longitude;
    float altitude;
    uint8_t satellites;
    uint8_t hour;
    uint8_t min;
    uint8_t sec;
    bool fix;
} gps_data_t;

// FUNCTIONS


extern void gps_init(void);

extern bool gps_update(void);

extern void gps_get_data(gps_data_t* data);

#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.

```
#ifndef MICROSD_MODULE_H
#define MICROSD_MODULE_H

#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"

extern bool sd_init();

extern void save_system_data(sensor_readings_t* sen_data, current_time_t* time_date);

//extern void save_photo(current_time_t* state, uint8_t* image_buffer, uint32_t length);

extern void save_gps_log(gps_data_t* gps);

#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.

```
#ifndef TIME_MANAGER_H
#define TIME_MANAGER_H

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

typedef struct
{
    uint16_t year;
    uint8_t month;
    uint8_t day;
    uint8_t hour;
    uint8_t min;
    uint8_t sec;
    uint16_t photo_count;
} current_time_t;

extern void time_manager_init(void);

extern bool time_manager_update(void);

extern void time_manager_sync(uint8_t hour, uint8_t min, uint8_t sec);

extern current_time_t* time_manager_get(void);

#endif
```