GROUP7 STM32 ENVIRONSENSING HAT

1.0

Generated by Doxygen 1.9.6

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 SENSEHAT INTEGRATION/main.c File Reference

: Main program body

```
#include "main.h"
#include "rtc.h"
#include "ldr.h"
#include "tmp102.h"
#include "EEPROM.h"
#include "string.h"
#include <math.h>
#include <stdbool.h>
```

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- int main (void)

  The application entry point.
- void read_and_store_data (void)

  Read and store data.
- void read_and_transmit_all_data (void)

  Read and transmit all data This function reads all the data stored in the EEPROM memory and transmits it over UART. It loops through all the pages of the EEPROM, reads the data from each page, and transmits it using UART communication. There is a delay between each UART transmission to ensure proper data transmission.
- void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)

  GPIO EXTI callback function This function is called when an EXTI interrupt event is triggered on a GPIO pin. It specifically handles the interrupt triggered by the PB10 pin, indicating that the USB has been plugged in. When this interrupt occurs, the function starts transmitting all the stored data over UART.
- void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef ∗htim)

  TIM period elapsed callback function.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

- ADC_HandleTypeDef hadc
- I2C_HandleTypeDef hi2c1
- RTC_HandleTypeDef hrtc
- TIM_HandleTypeDef htim2
- UART_HandleTypeDef huart1
- uint8_t hours = 07
- uint8_t minutes = 16
- uint8_t seconds = 0
- uint8_t day = 12
- uint8_t month = 5
- uint8_t year = 23
- uint8_t read_hours
- uint8_t read_minutes
- uint8_t read_seconds
- char time_string [9]
- uint8_t dayread
- uint8_t monthread
- uint8_t yearread
- char date_string [11]
- float light_intensity
- float temperature
- char temp_string [10]
- char light_string [10]
- HAL_StatusTypeDef status
- uint16_t current_page = 0
- bool eeprom_full = false
- bool usb_plugged = false

2.1.1 Detailed Description

: Main program body

Attention

Author

Rumbidzai Mashumba, Phomolo Makina, Travimadox Webb @company Imperium LLC

Date

13th of May 2023

2.1.2 Function Documentation

2.1.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

2.1.2.2 HAL_GPIO_EXTI_Callback()

```
void HAL_GPIO_EXTI_Callback (
            uint16_t GPIO_Pin )
```

GPIO EXTI callback function This function is called when an EXTI interrupt event is triggered on a GPIO pin. It specifically handles the interrupt triggered by the PB10 pin, indicating that the USB has been plugged in. When this interrupt occurs, the function starts transmitting all the stored data over UART.

Parameters

GPIO_Pin	The GPIO pin that triggered the interrupt

2.1.2.3 HAL_TIM_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback (
            TIM_HandleTypeDef * htim )
```

TIM period elapsed callback function.

This function is called when the period of a TIM (Timer) peripheral has elapsed. It specifically handles the callback for the timer used to track the elapsed time of 60 seconds. When this callback is triggered, the function reads data from the sensors and stores it.

2.1.2.4 main()

```
int main (
            void  )
```

The application entry point.

Return values

int	

2.1.2.5 read_and_store_data()

```
void read_and_store_data (
```

```
            void  )
```

Read and store data.

This function reads the light intensity, temperature, date, and time, and stores the data in the EEPROM memory.

The data is formatted using CSV (Comma-Separated Values) format and written to the EEPROM memory.

The function checks if the EEPROM is full before performing any write operations.

Note

If the EEPROM is full, the function does not perform any write operations.

2.1.2.6 read_and_transmit_all_data()

```
void read_and_transmit_all_data (
            void  )
```

Read and transmit all data This function reads all the data stored in the EEPROM memory and transmits it over UART. It loops through all the pages of the EEPROM, reads the data from each page, and transmits it using UART communication. There is a delay between each UART transmission to ensure proper data transmission.

Note

This function assumes that the EEPROM has been filled with data and the eeprom_full flag is not checked. If the EEPROM is not filled with data, this function may transmit garbage values.

2.1.2.7 SystemClock_Config()

```
void SystemClock_Config (
            void  )
```

System Clock Configuration.

Return values

None	

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

2.1.3 Variable Documentation

2.1.3.1 current_page

```
uint16_t current_page = 0
```

2.1.3.2 date_string

```
char date_string[11]
```

2.1.3.3 day

```
uint8_t day = 12
```

2.1.3.4 dayread

```
uint8_t dayread
```

2.1.3.5 eeprom_full

```
bool eeprom_full = false
```

2.1.3.6 hadc

```
ADC_HandleTypeDef hadc
```

2.1.3.7 hi2c1

```
I2C_HandleTypeDef hi2c1
```

2.1.3.8 hours

```
uint8_t hours = 07
```

2.1.3.9 hrtc

```
RTC_HandleTypeDef hrtc
```

2.1.3.10 htim2

```
TIM_HandleTypeDef htim2
```

2.1.3.11 huart1

```
UART_HandleTypeDef huart1
```

2.1.3.12 light_intensity

```
float light_intensity
```

2.1.3.13 light_string

```
char light_string[10]
```

2.1.3.14 minutes

```
uint8_t minutes = 16
```

2.1.3.15 month

```
uint8_t month = 5
```

2.1.3.16 monthread

```
uint8_t monthread
```

2.1.3.17 read_hours

```
uint8_t read_hours
```

2.1.3.18 read_minutes

```
uint8_t read_minutes
```

2.1.3.19 read_seconds

```
uint8_t read_seconds
```

2.1.3.20 seconds

```
uint8_t seconds = 0
```

2.1.3.21 status

```
HAL_StatusTypeDef status
```

2.1.3.22 temp_string

```
char temp_string[10]
```

2.1.3.23 temperature

```
float temperature
```

2.1.3.24 time_string

```
char time_string[9]
```

2.1.3.25 usb_plugged

```
bool usb_plugged = false
```

2.1.3.26 year

```
uint8_t year = 23
```

2.1.3.27 yearread

```
uint8_t yearread
```

2.2 SENSEHAT LIBRARIES/AT24C256/EEPROM.c File Reference

Using the HAL I2C Functions.

```
#include "EEPROM.h"
#include "math.h"
#include "string.h"
```

Macros

- #define EEPROM_I2C &hi2c1
- #define EEPROM_ADDR 0xA0
- #define PAGE_SIZE 64
- #define PAGE_NUM 512

Functions

- void EEPROM_Write (uint16_t page, uint16_t offset, uint8_t *data, uint16_t size)

  Write data to the EEPROM.
- void EEPROM_Write_NUM (uint16_t page, uint16_t offset, float data)

  Write a float/integer value to the EEPROM.
- float EEPROM_Read_NUM (uint16_t page, uint16_t offset)

  Read a single float/integer value from the EEPROM.
- void EEPROM_Read (uint16_t page, uint16_t offset, uint8_t *data, uint16_t size)

  Read data from the EEPROM.
- void EEPROM_PageErase (uint16_t page)

  Erase a page in the EEPROM Memory.

Variables

- uint8_t bytes_temp [4]
- I2C_HandleTypeDef hi2c1

2.2.1 Detailed Description

Using the HAL I2C Functions.

Author

ControllersTech

Date

Feb 16, 2021

Attention

2.2.2 Macro Definition Documentation

2.2.2.1 EEPROM_ADDR

```
#define EEPROM_ADDR 0xA0
```

2.2.2.2 EEPROM_I2C

```
#define EEPROM_I2C &hi2c1
```

2.2.2.3 PAGE_NUM

```
#define PAGE_NUM 512
```

2.2.2.4 PAGE_SIZE

```
#define PAGE_SIZE 64
```

2.2.3 Function Documentation

2.2.3.1 EEPROM_PageErase()

```
void EEPROM_PageErase (
            uint16_t page )
```

Erase a page in the EEPROM Memory.

Parameters

page	Number of page to erase In order to erase multiple pages, just use this function in the for loop

Return values

None	

2.2.3.2 EEPROM_Read()

```
void EEPROM_Read (
            uint16_t page,
            uint16_t offset,
            uint8_t * data,
            uint16_t size )
```

Read data from the EEPROM.

Parameters

page	Number of the start page. Range from 0 to PAGE_NUM-1.
offset	Start byte offset in the page. Range from 0 to PAGE_SIZE-1.
data	Pointer to the data to write in bytes.
size	Size of the data.

Return values

None.	

2.2.3.3 EEPROM_Read_NUM()

```
float EEPROM_Read_NUM (
            uint16_t page,
            uint16_t offset )
```

Read a single float/integer value from the EEPROM.

Read a float or integer value from the EEPROM.

Parameters

page	Number of the start page. Range from 0 to PAGE_NUM-1.
offset	Start byte offset in the page. Range from 0 to PAGE_SIZE-1.

Return values

Float/integer	value.

2.2.3.4 EEPROM_Write()

```
void EEPROM_Write (
            uint16_t page,
            uint16_t offset,
            uint8_t * data,
            uint16_t size )
```

Write data to the EEPROM.

Parameters

page	Start page number (0 to PAGE_NUM-1).
offset	Start byte offset in the page (0 to PAGE_SIZE-1).
data	Pointer to the data to write in bytes.
size	Size of the data.

Return values

None	

2.2.3.5 EEPROM_Write_NUM()

```
void EEPROM_Write_NUM (
            uint16_t page,
```

```
            uint16_t offset,
            float data )
```

Write a float/integer value to the EEPROM.

Write a float or integer value to the EEPROM.

Parameters

page	Number of the start page. Range from 0 to PAGE_NUM-1.
offset	Start byte offset in the page. Range from 0 to PAGE_SIZE-1.
data	Float/integer value that you want to write.

Return values

None.	

2.2.4 Variable Documentation

2.2.4.1 bytes_temp

```
uint8_t bytes_temp[4]
```

2.2.4.2 hi2c1

```
I2C_HandleTypeDef hi2c1  [extern]
```

2.3 SENSEHAT LIBRARIES/AT24C256/EEPROM.h File Reference

Using the HAL I2C Functions.

```
#include "stdint.h"
#include "stm32f0xx_hal.h"
```

Functions

- void EEPROM_Write (uint16_t page, uint16_t offset, uint8_t *data, uint16_t size)
  Write data to the EEPROM.
- void EEPROM_Read (uint16_t page, uint16_t offset, uint8_t *data, uint16_t size)
  Read data from the EEPROM.
- void EEPROM_PageErase (uint16_t page)
  Erase a page in the EEPROM Memory.
- void EEPROM_Write_NUM (uint16_t page, uint16_t offset, float fdata)
  Write a float or integer value to the EEPROM.
- float EEPROM_Read_NUM (uint16_t page, uint16_t offset)
  Read a float or integer value from the EEPROM.

2.3.1 Detailed Description

Using the HAL I2C Functions.

Author

ControllersTech

Date

Feb 16, 2021

Attention

2.3.2 Function Documentation

2.3.2.1 EEPROM_PageErase()

```
void EEPROM_PageErase (
            uint16_t page )
```

Erase a page in the EEPROM Memory.

Parameters

page	Page number to erase.

Return values

None	

Parameters

page	Number of page to erase In order to erase multiple pages, just use this function in the for loop

Return values

None	

2.3.2.2 EEPROM_Read()

```
void EEPROM_Read (
            uint16_t page,
            uint16_t offset,
            uint8_t * data,
            uint16_t size )
```

Read data from the EEPROM.

Parameters

page	Start page number (0 to PAGE_NUM-1).
offset	Start byte offset in the page (0 to PAGE_SIZE-1).
data	Pointer to the data to read in bytes.
size	Size of the data.

Return values

None	

Parameters

page	Number of the start page. Range from 0 to PAGE_NUM-1.
offset	Start byte offset in the page. Range from 0 to PAGE_SIZE-1.
data	Pointer to the data to write in bytes.
size	Size of the data.

Return values

None.	

2.3.2.3 EEPROM_Read_NUM()

```
float EEPROM_Read_NUM (
            uint16_t page,
            uint16_t offset )
```

Read a float or integer value from the EEPROM.

Parameters

page	Start page number (0 to PAGE_NUM-1).
offset	Start byte offset in the page (0 to PAGE_SIZE-1).

Return values

Float	or integer value read from the EEPROM.

Read a float or integer value from the EEPROM.

Parameters

page	Number of the start page. Range from 0 to PAGE_NUM-1.
offset	Start byte offset in the page. Range from 0 to PAGE_SIZE-1.

Return values

Float/integer	value.

2.3.2.4 EEPROM_Write()

```
void EEPROM_Write (
            uint16_t page,
            uint16_t offset,
            uint8_t * data,
            uint16_t size )
```

Write data to the EEPROM.

Parameters

page	Start page number (0 to PAGE_NUM-1).
offset	Start byte offset in the page (0 to PAGE_SIZE-1).
data	Pointer to the data to write in bytes.
size	Size of the data.

Return values

None	

2.3.2.5 EEPROM_Write_NUM()

```
void EEPROM_Write_NUM (
```

```
            uint16_t page,
            uint16_t offset,
            float data )
```

Write a float or integer value to the EEPROM.

Parameters

page	Start page number (0 to PAGE_NUM-1).
offset	Start byte offset in the page (0 to PAGE_SIZE-1).
fdata	Float or integer value to write.

Return values

None	

Write a float or integer value to the EEPROM.

Parameters

page	Number of the start page. Range from 0 to PAGE_NUM-1.
offset	Start byte offset in the page. Range from 0 to PAGE_SIZE-1.
data	Float/integer value that you want to write.

Return values

None.	

2.4 EEPROM.h

Go to the documentation of this file.

```
00001
00019 #ifndef INC_EEPROM_H_
00020 #define INC_EEPROM_H_
00021
00022 #include "stdint.h"
00023 #include "stm32f0xx_hal.h"
00024
00033 void EEPROM_Write(uint16_t page, uint16_t offset, uint8_t *data, uint16_t size);
00034
00043 void EEPROM_Read(uint16_t page, uint16_t offset, uint8_t *data, uint16_t size);
00044
00050 void EEPROM_PageErase(uint16_t page);
00051
00059 void EEPROM_Write_NUM(uint16_t page, uint16_t offset, float fdata);
00060
00067 float EEPROM_Read_NUM(uint16_t page, uint16_t offset);
00068
00069 #endif /* INC_EEPROM_H_ */
```

2.5 SENSEHAT LIBRARIES/LDR/ldr.c File Reference

LDR Library Source.

```
#include "ldr.h"
```

Functions

- void LDR_Init (void)

  Initialize the LDR with calibration constants.
- uint32_t LDR_ReadADC (ADC_HandleTypeDef ∗hadc)

  Read ADC value from the LDR.
- float LDR_ReadAnalogLightIntensity (ADC_HandleTypeDef ∗hadc)

  Read analog light intensity using the LDR.

2.5.1 Detailed Description

LDR Library Source.

Author

Travimadox Webb @position Embedded Software Engineer @company Imperium LLC

Date

6th of May 2023

2.5.2 Function Documentation

2.5.2.1 LDR_Init()

```
void LDR_Init (
            void  )
```

Initialize the LDR with calibration constants.

Initialize the LDR.

2.5.2.2 LDR_ReadADC()

```
uint32_t LDR_ReadADC (
            ADC_HandleTypeDef * hadc )
```

Read ADC value from the LDR.

Parameters

hadc	Pointer to an ADC_HandleTypeDef structure that contains the configuration information for the specified ADC

Returns

32-bit unsigned integer ADC value

2.5.2.3 LDR_ReadAnalogLightIntensity()

```
float LDR_ReadAnalogLightIntensity (
            ADC_HandleTypeDef * hadc )
```

Read analog light intensity using the LDR.

Parameters

hadc	Pointer to an ADC_HandleTypeDef structure that contains the configuration information for the specified ADC

Returns

Floating-point light intensity value

2.6 SENSEHAT LIBRARIES/LDR/ldr.h File Reference

LDR Library Header.

```
#include "stm32f0xx_hal.h"
```

Functions

- void LDR_Init (void)
  Initialize the LDR.
- uint32_t LDR_ReadADC (ADC_HandleTypeDef *hadc)
  Read ADC value from the LDR.
- float LDR_ReadAnalogLightIntensity (ADC_HandleTypeDef *hadc)
  Read analog light intensity using the LDR.

2.6.1 Detailed Description

LDR Library Header.

Author

Travimadox Webb @position Embedded Software Engineer @company Imperium LLC

Date

6th of May 2023

2.6.2 Function Documentation

2.6.2.1 LDR_Init()

```
void LDR_Init (
            void  )
```

Initialize the LDR.

Initialize the LDR.

2.6.2.2 LDR_ReadADC()

```
uint32_t LDR_ReadADC (
            ADC_HandleTypeDef * hadc )
```

Read ADC value from the LDR.

Parameters

hadc	Pointer to an ADC_HandleTypeDef structure that contains the configuration information for the specified ADC

Returns

32-bit unsigned integer ADC value

2.6.2.3 LDR_ReadAnalogLightIntensity()

```
float LDR_ReadAnalogLightIntensity (
            ADC_HandleTypeDef * hadc )
```

Read analog light intensity using the LDR.

Parameters

hadc	Pointer to an ADC_HandleTypeDef structure that contains the configuration information for the specified ADC

Returns

Floating-point light intensity value

2.7 ldr.h

Go to the documentation of this file.

```
00001
00010 #ifndef LDR_H
00011 #define LDR_H
00012
00013 #include "stm32f0xx_hal.h"
00014
00018 void LDR_Init(void);
00019
00025 uint32_t LDR_ReadADC(ADC_HandleTypeDef *hadc);
00026
00032 float LDR_ReadAnalogLightIntensity(ADC_HandleTypeDef *hadc);
00033
00034 #endif // LDR_H
```

2.8 SENSEHAT LIBRARIES/LTR303ALS/ltr303als.c File Reference

LTR-303ALS Ambient Light Sensor Library Implementation.

```
#include "ltr303als.h"
```

Functions

- HAL_StatusTypeDef LTR303ALS_Init (I2C_HandleTypeDef *hi2c, uint8_t integration_time, uint8_t measurement_rate)

  Initialize the LTR-303ALS Ambient Light Sensor.
- HAL_StatusTypeDef LTR303ALS_ReadLightIntensity (I2C_HandleTypeDef *hi2c, uint16_t *ch0, uint16_t *ch1)

  Read light intensity from the LTR-303ALS Ambient Light Sensor.

2.8.1 Detailed Description

LTR-303ALS Ambient Light Sensor Library Implementation.

Author

Travimadox Webb @position Embedded Software Engineer @company Imperium LLC

Date

6th of May 2023

2.8.2 Function Documentation

2.8.2.1 LTR303ALS_Init()

```
HAL_StatusTypeDef LTR303ALS_Init (
            I2C_HandleTypeDef * hi2c,
            uint8_t integration_time,
            uint8_t measurement_rate )
```

Initialize the LTR-303ALS Ambient Light Sensor.

Initialize the LTR-303ALS sensor.

Parameters

hi2c	Pointer to an I2C_HandleTypeDef structure that contains the configuration information for the specified I2C
integration_time	Integration time for the LTR-303ALS
measurement_rate	Measurement rate for the LTR-303ALS

Returns

HAL status (HAL_OK if successful, HAL_ERROR otherwise)

2.8.2.2 LTR303ALS_ReadLightIntensity()

```
HAL_StatusTypeDef LTR303ALS_ReadLightIntensity (
            I2C_HandleTypeDef * hi2c,
            uint16_t * ch0,
            uint16_t * ch1 )
```

Read light intensity from the LTR-303ALS Ambient Light Sensor.

Read light intensity data from the LTR-303ALS sensor.

Parameters

hi2c	Pointer to an I2C_HandleTypeDef structure that contains the configuration information for the specified I2C
ch0	Pointer to a uint16_t variable to store the Channel 0 data
ch1	Pointer to a uint16_t variable to store the Channel 1 data

Returns

HAL status (HAL_OK if successful, HAL_ERROR otherwise)

2.9 SENSEHAT LIBRARIES/LTR303ALS/ltr303als.h File Reference

LTR-303ALS Ambient Light Sensor Library Header.

```
#include "stm32f0xx_hal.h"
```

Macros

- #define LTR303ALS_I2C_ADDRESS 0x29

Functions

- HAL_StatusTypeDef LTR303ALS_Init (I2C_HandleTypeDef *hi2c, uint8_t integration_time, uint8_t measurement_rate)

  Initialize the LTR-303ALS sensor.
- HAL_StatusTypeDef LTR303ALS_ReadLightIntensity (I2C_HandleTypeDef *hi2c, uint16_t *ch0, uint16_t *ch1)

  Read light intensity data from the LTR-303ALS sensor.

2.9.1 Detailed Description

LTR-303ALS Ambient Light Sensor Library Header.

Author

Travimadox Webb @position Embedded Software Engineer @company Imperium LLC

Date

7th of May 2023

2.9.2 Macro Definition Documentation

2.9.2.1 LTR303ALS_I2C_ADDRESS

```
#define LTR303ALS_I2C_ADDRESS 0x29
```

2.9.3 Function Documentation

2.9.3.1 LTR303ALS_Init()

```
HAL_StatusTypeDef LTR303ALS_Init (
            I2C_HandleTypeDef * hi2c,
            uint8_t integration_time,
            uint8_t measurement_rate )
```

Initialize the LTR-303ALS sensor.

Parameters

hi2c	Pointer to the I2C handle.
integration_time	Integration time (in ms) for the sensor.
measurement_rate	Measurement rate (in ms) for the sensor.

Returns

HAL status.

Initialize the LTR-303ALS sensor.

Parameters

hi2c	Pointer to an I2C_HandleTypeDef structure that contains the configuration information for the specified I2C
integration_time	Integration time for the LTR-303ALS
measurement_rate	Measurement rate for the LTR-303ALS

Returns

HAL status (HAL_OK if successful, HAL_ERROR otherwise)

2.9.3.2 LTR303ALS_ReadLightIntensity()

```
HAL_StatusTypeDef LTR303ALS_ReadLightIntensity (
            I2C_HandleTypeDef * hi2c,
            uint16_t * ch0,
            uint16_t * ch1 )
```

Read light intensity data from the LTR-303ALS sensor.

Parameters

hi2c	Pointer to the I2C handle.
ch0	Pointer to store the channel 0 data.
ch1	Pointer to store the channel 1 data.

Returns

HAL status.

Read light intensity data from the LTR-303ALS sensor.

Parameters

hi2c	Pointer to an I2C_HandleTypeDef structure that contains the configuration information for the specified I2C
ch0	Pointer to a uint16_t variable to store the Channel 0 data
ch1	Pointer to a uint16_t variable to store the Channel 1 data

Returns

HAL status (HAL_OK if successful, HAL_ERROR otherwise)

2.10 ltr303als.h

Go to the documentation of this file.

```
00001
00010 #ifndef LTR303ALS_H
00011 #define LTR303ALS_H
00012
00013 #include "stm32f0xx_hal.h"
00014
00015 // LTR-303ALS I2C address (default: 0x29)
00016 #define LTR303ALS_I2C_ADDRESS 0x29
00017
00025 HAL_StatusTypeDef LTR303ALS_Init(I2C_HandleTypeDef *hi2c, uint8_t integration_time, uint8_t
      measurement_rate);
00026
00034 HAL_StatusTypeDef LTR303ALS_ReadLightIntensity(I2C_HandleTypeDef *hi2c, uint16_t *ch0, uint16_t *ch1);
00035
00036 #endif // LTR303ALS_H
```

2.11 SENSEHAT LIBRARIES/RTC/rtc.c File Reference

RTC Library Source.

```
#include "rtc.h"
```

Functions

- HAL_StatusTypeDef RTC_SetTime (RTC_HandleTypeDef *hrtc, uint8_t hours, uint8_t minutes, uint8_t seconds)

  Set the RTC time.
- HAL_StatusTypeDef RTC_GetTime (RTC_HandleTypeDef *hrtc, uint8_t *hours, uint8_t *minutes, uint8_t *seconds)

  Get the RTC time.
- void RTC_SetDate (RTC_HandleTypeDef *hrtc, uint8_t day, uint8_t month, uint8_t year)

  Set the RTC date.
- void RTC_GetDate (RTC_HandleTypeDef *hrtc, uint8_t *day, uint8_t *month, uint8_t *year)

  Get the RTC date.

2.11.1 Detailed Description

RTC Library Source.

Author

Travimadox Webb @position Embedded Software Engineer @company Imperium LLC

Date

7th May 2023

Note

IMPORTANT: Initialize the RTC module using STM32CubeIDE GUI. Configure RTC clock source and enable RTC in the main initialization code or in the Clock Configuration function (SystemClock_Config).

2.11.2 Function Documentation

2.11.2.1 RTC_GetDate()

```
void RTC_GetDate (
            RTC_HandleTypeDef * hrtc,
            uint8_t * day,
            uint8_t * month,
            uint8_t * year )
```

Get the RTC date.

Parameters

hrtc	Pointer to an RTC_HandleTypeDef structure that contains the configuration information for the specified RTC
day	Pointer to store the day of the month
month	Pointer to store the month
year	Pointer to store the year

2.11.2.2 RTC_GetTime()

```
HAL_StatusTypeDef RTC_GetTime (
            RTC_HandleTypeDef * hrtc,
            uint8_t * hours,
            uint8_t * minutes,
            uint8_t * seconds )
```

Get the RTC time.

Parameters

hrtc	Pointer to an RTC_HandleTypeDef structure that contains the configuration information for the specified RTC
hours	Pointer to an uint8_t variable to store the hours value
minutes	Pointer to an uint8_t variable to store the minutes value
seconds	Pointer to an uint8_t variable to store the seconds value

Returns

HAL status (HAL_OK, HAL_ERROR, HAL_BUSY, or HAL_TIMEOUT)

2.11.2.3 RTC_SetDate()

```
void RTC_SetDate (
             RTC_HandleTypeDef * hrtc,
             uint8_t day,
             uint8_t month,
             uint8_t year )
```

Set the RTC date.

Parameters

hrtc	Pointer to an RTC_HandleTypeDef structure that contains the configuration information for the specified RTC
day	The day of the month
month	The month
year	The year (from 0 to 99)

2.11.2.4 RTC_SetTime()

```
HAL_StatusTypeDef RTC_SetTime (
             RTC_HandleTypeDef * hrtc,
             uint8_t hours,
             uint8_t minutes,
             uint8_t seconds )
```

Set the RTC time.

Parameters

hrtc	Pointer to an RTC_HandleTypeDef structure that contains the configuration information for the specified RTC
hours	Hours value to set (0-23)
minutes	Minutes value to set (0-59)
seconds	Seconds value to set (0-59)

Returns

HAL status (HAL_OK, HAL_ERROR, HAL_BUSY, or HAL_TIMEOUT)

2.12 SENSEHAT LIBRARIES/RTC/rtc.h File Reference

RTC Library Header.

```
#include "stm32f0xx_hal.h"
```

Functions

- HAL_StatusTypeDef RTC_SetTime (RTC_HandleTypeDef *hrtc, uint8_t hours, uint8_t minutes, uint8_t seconds)

  Set the RTC time.
- HAL_StatusTypeDef RTC_GetTime (RTC_HandleTypeDef *hrtc, uint8_t *hours, uint8_t *minutes, uint8_t *seconds)

  Get the RTC time.
- void RTC_SetDate (RTC_HandleTypeDef *hrtc, uint8_t day, uint8_t month, uint8_t year)

  Set the RTC date.
- void RTC_GetDate (RTC_HandleTypeDef *hrtc, uint8_t *day, uint8_t *month, uint8_t *year)

  Get the RTC date.

2.12.1 Detailed Description

RTC Library Header.

Author

Travimadox Webb @position Embedded Software Engineer @company Imperium LLC

Date

7th May 2023

2.12.2 Function Documentation

2.12.2.1 RTC_GetDate()

```
void RTC_GetDate (
            RTC_HandleTypeDef * hrtc,
            uint8_t * day,
            uint8_t * month,
            uint8_t * year )
```

Get the RTC date.

Parameters

hrtc	Pointer to an RTC_HandleTypeDef structure that contains the configuration information for the specified RTC
day	Pointer to store the day of the month
month	Pointer to store the month
year	Pointer to store the year

2.12.2.2 RTC_GetTime()

```
HAL_StatusTypeDef RTC_GetTime (
            RTC_HandleTypeDef * hrtc,
            uint8_t * hours,
            uint8_t * minutes,
            uint8_t * seconds )
```

Get the RTC time.

Parameters

hrtc	Pointer to an RTC_HandleTypeDef structure that contains the configuration information for the specified RTC
hours	Pointer to an uint8_t variable to store the hours value
minutes	Pointer to an uint8_t variable to store the minutes value
seconds	Pointer to an uint8_t variable to store the seconds value

Returns

HAL status (HAL_OK, HAL_ERROR, HAL_BUSY, or HAL_TIMEOUT)

2.12.2.3 RTC_SetDate()

```
void RTC_SetDate (
            RTC_HandleTypeDef * hrtc,
            uint8_t day,
            uint8_t month,
            uint8_t year )
```

Set the RTC date.

Parameters

hrtc	Pointer to an RTC_HandleTypeDef structure that contains the configuration information for the specified RTC
day	The day of the month
month	The month
year	The year (from 0 to 99)

2.12.2.4 RTC_SetTime()

```
HAL_StatusTypeDef RTC_SetTime (
            RTC_HandleTypeDef * hrtc,
            uint8_t hours,
```

```
            uint8_t minutes,
            uint8_t seconds )
```

Set the RTC time.

Parameters

hrtc	Pointer to an RTC_HandleTypeDef structure that contains the configuration information for the specified RTC
hours	Hours value to set (0-23)
minutes	Minutes value to set (0-59)
seconds	Seconds value to set (0-59)

Returns

HAL status (HAL_OK, HAL_ERROR, HAL_BUSY, or HAL_TIMEOUT)

2.13 rtc.h

Go to the documentation of this file.

```
00001
00010 #ifndef __RTC_H
00011 #define __RTC_H
00012
00013 #include "stm32f0xx_hal.h"
00014
00023 HAL_StatusTypeDef RTC_SetTime(RTC_HandleTypeDef *hrtc, uint8_t hours, uint8_t minutes, uint8_t
      seconds);
00024
00033 HAL_StatusTypeDef RTC_GetTime(RTC_HandleTypeDef *hrtc, uint8_t *hours, uint8_t *minutes, uint8_t
      *seconds);
00034
00035
00044 void RTC_SetDate(RTC_HandleTypeDef *hrtc, uint8_t day, uint8_t month, uint8_t year);
00053 void RTC_GetDate(RTC_HandleTypeDef *hrtc, uint8_t *day, uint8_t *month, uint8_t *year);
00054
00055 #endif // __RTC_H
```

2.14 SENSEHAT LIBRARIES/TMP102/tmp102.c File Reference

TMP102 Temperature Sensor Library.

```
#include "tmp102.h"
#include <math.h>
```

Functions

- HAL_StatusTypeDef TMP102_Init (I2C_HandleTypeDef *hi2c)

  Initialize the TMP102 temperature sensor.
- float TMP102_ReadTemperature (I2C_HandleTypeDef *hi2c)

  Read temperature from the TMP102 sensor.

2.14.1 Detailed Description

TMP102 Temperature Sensor Library.

Author

Travimadox Webb @position Embedded Software Engineer @company Imperium LLC

Date

6th of May 2023

2.14.2 Function Documentation

2.14.2.1 TMP102_Init()

```
HAL_StatusTypeDef TMP102_Init (
            I2C_HandleTypeDef * hi2c )
```

Initialize the TMP102 temperature sensor.

Parameters

hi2c	Pointer to an I2C_HandleTypeDef structure that contains the configuration information for the specified I2C peripheral.

Return values

HAL	status

2.14.2.2 TMP102_ReadTemperature()

```
float TMP102_ReadTemperature (
            I2C_HandleTypeDef * hi2c )
```

Read temperature from the TMP102 sensor.

Parameters

hi2c	Pointer to an I2C_HandleTypeDef structure that contains the configuration information for the specified I2C peripheral.

Returns

Temperature in degrees Celsius as a float.

2.15 SENSEHAT LIBRARIES/TMP102/tmp102.h File Reference

```
#include "stm32f0xx_hal.h"
```

Macros

- #define TMP102_I2C_ADDRESS 0x48
- #define TMP102_REG_TEMPERATURE 0x00
- #define TMP102_REG_CONFIG 0x01
- #define TMP102_CONFIG_CONTINUOUS_CONVERSION 0x0000
- #define TMP102_CONFIG_SHUTDOWN_MODE 0x0100

Functions

- HAL_StatusTypeDef TMP102_Init (I2C_HandleTypeDef *hi2c)

  Initialize the TMP102 temperature sensor.
- float TMP102_ReadTemperature (I2C_HandleTypeDef *hi2c)

  Read temperature from the TMP102 sensor.

2.15.1 Macro Definition Documentation

2.15.1.1 TMP102_CONFIG_CONTINUOUS_CONVERSION

```
#define TMP102_CONFIG_CONTINUOUS_CONVERSION 0x0000
```

2.15.1.2 TMP102_CONFIG_SHUTDOWN_MODE

```
#define TMP102_CONFIG_SHUTDOWN_MODE 0x0100
```

2.15.1.3 TMP102_I2C_ADDRESS

```
#define TMP102_I2C_ADDRESS 0x48
```

2.15.1.4 TMP102_REG_CONFIG

```
#define TMP102_REG_CONFIG 0x01
```

2.15.1.5 TMP102_REG_TEMPERATURE

```
#define TMP102_REG_TEMPERATURE 0x00
```

2.15.2 Function Documentation

2.15.2.1 TMP102_Init()

```
HAL_StatusTypeDef TMP102_Init (
            I2C_HandleTypeDef * hi2c )
```

Initialize the TMP102 temperature sensor.

Parameters

hi2c	Pointer to an I2C_HandleTypeDef structure that contains the configuration information for the specified I2C peripheral.

Return values

HAL	status

2.15.2.2 TMP102_ReadTemperature()

```
float TMP102_ReadTemperature (
            I2C_HandleTypeDef * hi2c )
```

Read temperature from the TMP102 sensor.

Parameters

hi2c	Pointer to an I2C_HandleTypeDef structure that contains the configuration information for the specified I2C peripheral.

Returns

Temperature in degrees Celsius as a float.

2.16 tmp102.h

Go to the documentation of this file.

```
00001 /*
00002  * TMP102 Temperature Sensor Library
00003  * Author: Travimadox Webb
00004  * Postion: Embedded Software Engineer
00005  * Company: Imperium LLC
00006  * Date: 6th of May 2023
00007  */
00008
00009 #ifndef TMP102_H
00010 #define TMP102_H
00011
00012 #include "stm32f0xx_hal.h"
00013
00014 // TMP102 I2C address (default: 0x48)
00015 #define TMP102_I2C_ADDRESS 0x48
00016
00017 // TMP102 register addresses
00018 #define TMP102_REG_TEMPERATURE 0x00
00019 #define TMP102_REG_CONFIG 0x01
00020
00021 // TMP102 configuration settings
00022 #define TMP102_CONFIG_CONTINUOUS_CONVERSION 0x0000
00023 #define TMP102_CONFIG_SHUTDOWN_MODE 0x0100
00024
00025 // Initialize the TMP102 sensor
00026 HAL_StatusTypeDef TMP102_Init(I2C_HandleTypeDef *hi2c);
00027
00028 // Read temperature from the TMP102 sensor
00029 float TMP102_ReadTemperature(I2C_HandleTypeDef *hi2c);
00030
00031 #endif // TMP102_H
```

Index