subsubsubsection4 subsubsubsubsection5 subsubsubsubsubsection6 subsubsubsubsubsubsection7 paragraph8 subparagraph9

bmp280-stm32f1

Generated by Doxygen 1.9.7

Contents

Chapter 1

bmp280-stm32f1

Simple BMP280 driver embedded in example configured STM32CubeIDE project for STM32F103.

It should work with any STM32 MCU after replacing ¨#include ¨stm32f1xx_hal.h¨¨ in BMP280.h with correct one.

1.1 Wiring

For this code to run ¨out of the box¨ you need an STM32F103 and BMP280 sensor connected as follows:

Sensor<->MCU

VCC <-> 3V3

GND<->GND

SCL<->B6 (I2C1 clock)

SDA<->B7 (I2C1 data)

CSB<->3V3 (Disable SPI on sensor)

SDD<->GND (Set sensor address to 0x76)

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 BMP280_Result Struct Reference

`#include <BMP280.h>`

Data Fields

- float Temperature
- float Pressure

4.1.1 Field Documentation

4.1.1.1 Pressure

`float BMP280_Result::Pressure`

4.1.1.2 Temperature

`float BMP280_Result::Temperature`

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

- Core/Inc/BMP280.h

Chapter 5

File Documentation

5.1 Core/Inc/BMP280.h File Reference

BMP280 driver file header.

```
#include ¨stm32f1xx_hal.h¨
#include <stdbool.h>
```

Data Structures

- struct BMP280_Result

Macros

Sensor I2C addresses

- #define BMP280_DEVICE_ADDRESS_GND (0x76 << 1)
- #define BMP280_DEVICE_ADDRESS_VDDIO (0x77 << 1)

Raw temperature data registers addresses, read only

- #define BMP280_REG_TEMP_XLSB 0xFC
- #define BMP280_REG_TEMP_LSB 0xFB
- #define BMP280_REG_TEMP_MSB 0xFA

Raw pressure data registers addresses, read only

- #define BMP280_REG_PRESS_XLSB 0xF9
- #define BMP280_REG_PRESS_LSB 0xF8
- #define BMP280_REG_PRESS_MSB 0xF7

Control registers addresses, read+write

- #define BMP280_REG_CONFIG 0xF5
- #define BMP280_REG_CTRL_MEAS 0xF4

Status register address, do not write

- #define BMP280_REG_STATUS 0xF3

Reset register address, write only

- #define BMP280_REG_RESET 0xE0

Device ID register address, read only

- #define BMP280_REG_ID 0xD0

Calibration registers addresses, read only

- #define BMP280_REG_CALIB25 0xA1
- #define BMP280_REG_CALIB24 0xA0
- #define BMP280_REG_CALIB23 0x9F
- #define BMP280_REG_CALIB22 0x9E
- #define BMP280_REG_CALIB21 0x9D
- #define BMP280_REG_CALIB20 0x9C
- #define BMP280_REG_CALIB19 0x9B
- #define BMP280_REG_CALIB18 0x9A
- #define BMP280_REG_CALIB17 0x99
- #define BMP280_REG_CALIB16 0x98
- #define BMP280_REG_CALIB15 0x97
- #define BMP280_REG_CALIB14 0x96
- #define BMP280_REG_CALIB13 0x95
- #define BMP280_REG_CALIB12 0x94
- #define BMP280_REG_CALIB11 0x93
- #define BMP280_REG_CALIB10 0x92
- #define BMP280_REG_CALIB09 0x91
- #define BMP280_REG_CALIB08 0x90
- #define BMP280_REG_CALIB07 0x8F
- #define BMP280_REG_CALIB06 0x8E
- #define BMP280_REG_CALIB05 0x8D
- #define BMP280_REG_CALIB04 0x8C
- #define BMP280_REG_CALIB03 0x8B
- #define BMP280_REG_CALIB02 0x8A
- #define BMP280_REG_CALIB01 0x89
- #define BMP280_REG_CALIB00 0x88

Valid ID register value

- #define BMP280_VAL_DEVID 0x58

Valid RESET register value

- #define BMP280_VAL_RESET 0xB6

Valid STATUS register flags masks

- #define BMP280_VAL_STATUS_MEASURING 1U << 3
- #define BMP280_VAL_STATUS_IM_UPDATE 1U << 0

Valid CTRL_MEAS register acquisition option values

- #define BMP280_VAL_CTRL_MEAS_OSRS_T_0 0x00
- #define BMP280_VAL_CTRL_MEAS_OSRS_T_1 0x01

- #define BMP280_VAL_CTRL_MEAS_OSRS_T_2 0x02
- #define BMP280_VAL_CTRL_MEAS_OSRS_T_4 0x03
- #define BMP280_VAL_CTRL_MEAS_OSRS_T_8 0x04
- #define BMP280_VAL_CTRL_MEAS_OSRS_T_16 0x05
- #define BMP280_VAL_CTRL_MEAS_OSRS_P_0 0x00
- #define BMP280_VAL_CTRL_MEAS_OSRS_P_1 0x01
- #define BMP280_VAL_CTRL_MEAS_OSRS_P_2 0x02
- #define BMP280_VAL_CTRL_MEAS_OSRS_P_4 0x03
- #define BMP280_VAL_CTRL_MEAS_OSRS_P_8 0x04
- #define BMP280_VAL_CTRL_MEAS_OSRS_P_16 0x05
- #define BMP280_VAL_CTRL_MEAS_MODE_SLEEP 0x00
- #define BMP280_VAL_CTRL_MEAS_MODE_FORCED 0x01
- #define BMP280_VAL_CTRL_MEAS_MODE_NORMAL 0x03
- #define BMP280_VAL_CTRL_CONFIG_T_SB_0_5 0x00
- #define BMP280_VAL_CTRL_CONFIG_T_SB_62_5 0x01
- #define BMP280_VAL_CTRL_CONFIG_T_SB_125 0x02
- #define BMP280_VAL_CTRL_CONFIG_T_SB_250 0x03
- #define BMP280_VAL_CTRL_CONFIG_T_SB_500 0x04
- #define BMP280_VAL_CTRL_CONFIG_T_SB_1000 0x05
- #define BMP280_VAL_CTRL_CONFIG_T_SB_2000 0x06
- #define BMP280_VAL_CTRL_CONFIG_T_SB_4000 0x07
- #define BMP280_VAL_CTRL_CONFIG_FILTER_0 0x00
- #define BMP280_VAL_CTRL_CONFIG_FILTER_2 0x01
- #define BMP280_VAL_CTRL_CONFIG_FILTER_4 0x02
- #define BMP280_VAL_CTRL_CONFIG_FILTER_8 0x03
- #define BMP280_VAL_CTRL_CONFIG_FILTER_16 0x04
- #define BMP280_VAL_CTRL_SPI3W_EN 0b00000001

MCU specific setting - affects pressure processing formula

- #define RETURN_64BIT true
- #define RETURN_32BIT false

Typedefs

- typedef struct BMP280_Result BMP280_Result

Functions

- bool BMP280_Init_I2C (uint8_t osrs_t, uint8_t osrs_p, uint8_t acq_mode, uint8_t t_sb, uint8_t filter_tc, I2C↩ _HandleTypeDef i2c_handle, uint8_t device_address)

  Initialize sensor with chosen settings.
- void BMP280_CalibrationConstantsRead_I2C (I2C_HandleTypeDef i2c_handle, uint8_t device_address)

  Read sensor calibration parameters over I2C.
- bool BMP280_Wake_I2C (I2C_HandleTypeDef i2c_handle, uint8_t device_address)

  Wake sensor over I2C - used when measuring in forced mode.
- struct BMP280_Result BMP280_Measure_I2C (I2C_HandleTypeDef i2c_handle, uint8_t device_address)

  Measure temperature and pressure over I2C.

5.1.1 Detailed Description

BMP280 driver file header.

Created on: Apr 5, 2024
Author: Piotr Jucha

5.1.2 Macro Definition Documentation

5.1.2.1 BMP280_DEVICE_ADDRESS_GND

```
#define BMP280_DEVICE_ADDRESS_GND  (0x76 << 1)
```

Used when SDO pulled down \

5.1.2.2 BMP280_DEVICE_ADDRESS_VDDIO

```
#define BMP280_DEVICE_ADDRESS_VDDIO  (0x77 << 1)
```

Used when SDO pulled up \

5.1.2.3 BMP280_REG_CALIB00

```
#define BMP280_REG_CALIB00 0x88
```

Calibration register 0

5.1.2.4 BMP280_REG_CALIB01

```
#define BMP280_REG_CALIB01 0x89
```

Calibration register 1

5.1.2.5 BMP280_REG_CALIB02

```
#define BMP280_REG_CALIB02 0x8A
```

Calibration register 2

5.1.2.6 BMP280_REG_CALIB03

```
#define BMP280_REG_CALIB03 0x8B
```

Calibration register 3

5.1.2.7 BMP280_REG_CALIB04

```
#define BMP280_REG_CALIB04 0x8C
```

Calibration register 4

5.1.2.8 BMP280_REG_CALIB05

```
#define BMP280_REG_CALIB05 0x8D
```

Calibration register 5

5.1.2.9 BMP280_REG_CALIB06

```
#define BMP280_REG_CALIB06 0x8E
```

Calibration register 6

5.1.2.10 BMP280_REG_CALIB07

```
#define BMP280_REG_CALIB07 0x8F
```

Calibration register 7

5.1.2.11 BMP280_REG_CALIB08

```
#define BMP280_REG_CALIB08 0x90
```

Calibration register 8

5.1.2.12 BMP280_REG_CALIB09

```
#define BMP280_REG_CALIB09 0x91
```

Calibration register 9

5.1.2.13 BMP280_REG_CALIB10

```
#define BMP280_REG_CALIB10 0x92
```

Calibration register 10

5.1.2.14 BMP280_REG_CALIB11

```
#define BMP280_REG_CALIB11 0x93
```

Calibration register 11

5.1.2.15 BMP280_REG_CALIB12

```
#define BMP280_REG_CALIB12 0x94
```

Calibration register 12

5.1.2.16 BMP280_REG_CALIB13

```
#define BMP280_REG_CALIB13 0x95
```

Calibration register 13

5.1.2.17 BMP280_REG_CALIB14

```
#define BMP280_REG_CALIB14 0x96
```

Calibration register 14

5.1.2.18 BMP280_REG_CALIB15

```
#define BMP280_REG_CALIB15 0x97
```

Calibration register 15

5.1.2.19 BMP280_REG_CALIB16

```
#define BMP280_REG_CALIB16 0x98
```

Calibration register 16

5.1.2.20 BMP280_REG_CALIB17

```
#define BMP280_REG_CALIB17 0x99
```

Calibration register 17

5.1.2.21 BMP280_REG_CALIB18

```
#define BMP280_REG_CALIB18 0x9A
```

Calibration register 18

5.1.2.22 BMP280_REG_CALIB19

```
#define BMP280_REG_CALIB19 0x9B
```

Calibration register 19

5.1.2.23 BMP280_REG_CALIB20

```
#define BMP280_REG_CALIB20 0x9C
```

Calibration register 20

5.1.2.24 BMP280_REG_CALIB21

```
#define BMP280_REG_CALIB21 0x9D
```

Calibration register 21

5.1.2.25 BMP280_REG_CALIB22

```
#define BMP280_REG_CALIB22 0x9E
```

Calibration register 22

5.1.2.26 BMP280_REG_CALIB23

```
#define BMP280_REG_CALIB23 0x9F
```

Calibration register 23

5.1.2.27 BMP280_REG_CALIB24

```
#define BMP280_REG_CALIB24 0xA0
```

Calibration register 24

5.1.2.28 BMP280_REG_CALIB25

```
#define BMP280_REG_CALIB25 0xA1
```

Calibration register 25

5.1.2.29 BMP280_REG_CONFIG

```
#define BMP280_REG_CONFIG 0xF5
```

Configuration register

5.1.2.30 BMP280_REG_CTRL_MEAS

```
#define BMP280_REG_CTRL_MEAS 0xF4
```

Data acquisition mode register

5.1.2.31 BMP280_REG_ID

```
#define BMP280_REG_ID 0xD0
```

DEVID register

5.1.2.32 BMP280_REG_PRESS_LSB

```
#define BMP280_REG_PRESS_LSB 0xF8
```

Middle pressure data chunk

5.1.2.33 BMP280_REG_PRESS_MSB

```
#define BMP280_REG_PRESS_MSB 0xF7
```

MSB pressure data chunk

5.1.2.34 BMP280_REG_PRESS_XLSB

```
#define BMP280_REG_PRESS_XLSB 0xF9
```

LSB pressure data chunk

5.1.2.35 BMP280_REG_RESET

```
#define BMP280_REG_RESET 0xE0
```

Device reset register

5.1.2.36 BMP280_REG_STATUS

```
#define BMP280_REG_STATUS 0xF3
```

Device status register

5.1.2.37 BMP280_REG_TEMP_LSB

```
#define BMP280_REG_TEMP_LSB 0xFB
```

Middle temperature data chunk

5.1.2.38 BMP280_REG_TEMP_MSB

```
#define BMP280_REG_TEMP_MSB 0xFA
```

MSB temperature data chunk

5.1.2.39 BMP280_REG_TEMP_XLSB

```
#define BMP280_REG_TEMP_XLSB 0xFC
```

LSB temperature data chunk

5.1.2.40 BMP280_VAL_CTRL_CONFIG_FILTER_0

```
#define BMP280_VAL_CTRL_CONFIG_FILTER_0 0x00
```

Filter disabled

5.1.2.41 BMP280_VAL_CTRL_CONFIG_FILTER_16

```
#define BMP280_VAL_CTRL_CONFIG_FILTER_16 0x04
```

16x filter

5.1.2.42 BMP280_VAL_CTRL_CONFIG_FILTER_2

```
#define BMP280_VAL_CTRL_CONFIG_FILTER_2 0x01
```

2x filter

5.1.2.43 BMP280_VAL_CTRL_CONFIG_FILTER_4

```
#define BMP280_VAL_CTRL_CONFIG_FILTER_4 0x02
```

4x filter

5.1.2.44 BMP280_VAL_CTRL_CONFIG_FILTER_8

```
#define BMP280_VAL_CTRL_CONFIG_FILTER_8 0x03
```

8x filter

5.1.2.45 BMP280_VAL_CTRL_CONFIG_T_SB_0_5

```
#define BMP280_VAL_CTRL_CONFIG_T_SB_0_5 0x00
```

Standby 0.5ms

5.1.2.46 BMP280_VAL_CTRL_CONFIG_T_SB_1000

```
#define BMP280_VAL_CTRL_CONFIG_T_SB_1000 0x05
```

Standby 1000ms

5.1.2.47 BMP280_VAL_CTRL_CONFIG_T_SB_125

```
#define BMP280_VAL_CTRL_CONFIG_T_SB_125 0x02
```

Standby 125ms

5.1.2.48 BMP280_VAL_CTRL_CONFIG_T_SB_2000

```
#define BMP280_VAL_CTRL_CONFIG_T_SB_2000 0x06
```

Standby 2000ms

5.1.2.49 BMP280_VAL_CTRL_CONFIG_T_SB_250

```
#define BMP280_VAL_CTRL_CONFIG_T_SB_250 0x03
```

Standby 250ms

5.1.2.50 BMP280_VAL_CTRL_CONFIG_T_SB_4000

```
#define BMP280_VAL_CTRL_CONFIG_T_SB_4000 0x07
```

Standby 4000ms

5.1.2.51 BMP280_VAL_CTRL_CONFIG_T_SB_500

```
#define BMP280_VAL_CTRL_CONFIG_T_SB_500 0x04
```

Standby 500ms

5.1.2.52 BMP280_VAL_CTRL_CONFIG_T_SB_62_5

```
#define BMP280_VAL_CTRL_CONFIG_T_SB_62_5 0x01
```

Standby 62.25ms

5.1.2.53 BMP280_VAL_CTRL_MEAS_MODE_FORCED

```
#define BMP280_VAL_CTRL_MEAS_MODE_FORCED 0x01
```

Forced mode

5.1.2.54 BMP280_VAL_CTRL_MEAS_MODE_NORMAL

```
#define BMP280_VAL_CTRL_MEAS_MODE_NORMAL 0x03
```

Continuous mode

5.1.2.55 BMP280_VAL_CTRL_MEAS_MODE_SLEEP

```
#define BMP280_VAL_CTRL_MEAS_MODE_SLEEP 0x00
```

Sleep mode

5.1.2.56 BMP280_VAL_CTRL_MEAS_OSRS_P_0

```
#define BMP280_VAL_CTRL_MEAS_OSRS_P_0 0x00
```

Disabled measurement

5.1.2.57 BMP280_VAL_CTRL_MEAS_OSRS_P_1

```
#define BMP280_VAL_CTRL_MEAS_OSRS_P_1 0x01
```

1x oversampling

5.1.2.58 BMP280_VAL_CTRL_MEAS_OSRS_P_16

```
#define BMP280_VAL_CTRL_MEAS_OSRS_P_16 0x05
```

16x oversampling

5.1.2.59 BMP280_VAL_CTRL_MEAS_OSRS_P_2

```
#define BMP280_VAL_CTRL_MEAS_OSRS_P_2 0x02
```

2x oversampling

5.1.2.60 BMP280_VAL_CTRL_MEAS_OSRS_P_4

```
#define BMP280_VAL_CTRL_MEAS_OSRS_P_4 0x03
```

4x oversampling

5.1.2.61 BMP280_VAL_CTRL_MEAS_OSRS_P_8

```
#define BMP280_VAL_CTRL_MEAS_OSRS_P_8 0x04
```

8x oversampling

5.1.2.62 BMP280_VAL_CTRL_MEAS_OSRS_T_0

```
#define BMP280_VAL_CTRL_MEAS_OSRS_T_0 0x00
```

Disabled measurement

5.1.2.63 BMP280_VAL_CTRL_MEAS_OSRS_T_1

```
#define BMP280_VAL_CTRL_MEAS_OSRS_T_1 0x01
```

1x oversampling

5.1.2.64 BMP280_VAL_CTRL_MEAS_OSRS_T_16

```
#define BMP280_VAL_CTRL_MEAS_OSRS_T_16 0x05
```

16x oversampling

5.1.2.65 BMP280_VAL_CTRL_MEAS_OSRS_T_2

```
#define BMP280_VAL_CTRL_MEAS_OSRS_T_2 0x02
```

2x oversampling

5.1.2.66 BMP280_VAL_CTRL_MEAS_OSRS_T_4

```
#define BMP280_VAL_CTRL_MEAS_OSRS_T_4 0x03
```

4x oversampling

5.1.2.67 BMP280_VAL_CTRL_MEAS_OSRS_T_8

```
#define BMP280_VAL_CTRL_MEAS_OSRS_T_8 0x04
```

8x oversampling

5.1.2.68 BMP280_VAL_CTRL_SPI3W_EN

```
#define BMP280_VAL_CTRL_SPI3W_EN 0b00000001
```

Enable SPI 3-wire

5.1.2.69 BMP280_VAL_DEVID

```
#define BMP280_VAL_DEVID 0x58
```

Device ID

5.1.2.70 BMP280_VAL_RESET

```
#define BMP280_VAL_RESET 0xB6
```

Writing to reset register resets device

5.1.2.71 BMP280_VAL_STATUS_IM_UPDATE

```
#define BMP280_VAL_STATUS_IM_UPDATE 1U << 0
```

NVM copying status flag

5.1.2.72 BMP280_VAL_STATUS_MEASURING

```
#define BMP280_VAL_STATUS_MEASURING 1U << 3
```

Measuring busy flag

5.1.2.73 RETURN_32BIT

```
#define RETURN_32BIT false
```

for MCU without 64-bit operations support

5.1.2.74 RETURN_64BIT

```
#define RETURN_64BIT true
```

for MCU with 64-bit operations support

5.1.3 Typedef Documentation

5.1.3.1 BMP280_Result

```
typedef struct BMP280_Result BMP280_Result
```

5.1.4 Function Documentation

5.1.4.1 BMP280_CalibrationConstantsRead_I2C()

```
void BMP280_CalibrationConstantsRead_I2C (
            I2C_HandleTypeDef i2c_handle,
            uint8_t device_address )
```

Read sensor calibration parameters over I2C.

Parameters

i2c_handle	Desired MCU I2C peripheral for communication with sensor
device_address	I2C device address

Read constants used for temperature and pressure calculations from sensor's memory

5.1.4.2 BMP280_Init_I2C()

```
bool BMP280_Init_I2C (
            uint8_t osrs_t,
```

```
            uint8_t osrs_p,
            uint8_t acq_mode,
            uint8_t t_sb,
            uint8_t filter_tc,
            I2C_HandleTypeDef i2c_handle,
            uint8_t device_address )
```

Initialize sensor with chosen settings.

Parameters

osrs_t	Temperature oversampling setting
osrs_p	Pressure oversampling setting
acq_mode	Acquisition mode setting
t_sb	Standby time setting
filter_tc	Sensor IIR Filter time constant setting
i2c_handle	Desired MCU I2C peripheral for communication with sensor
device_address	I2C device address

Returns

Configuration status
false == unsuccessful
true == successful

Reset the sensor, check if device ID is valid, read calibration constants, write oversampling, acquisition mode, readout timing and filter data to the sensor

5.1.4.3 BMP280_Measure_I2C()

```
struct BMP280_Result BMP280_Measure_I2C (
            I2C_HandleTypeDef i2c_handle,
            uint8_t device_address )
```

Measure temperature and pressure over I2C.

Parameters

i2c_handle	Desired MCU I2C peripheral for communication with sensor
device_address	I2C device address BMP280_DEVICE_ADDRESS_GND = 0x76 BMP280_DEVICE_ADDRESS_VDDIO = 0x77

Returns

Measurement values

5.1.4.4 BMP280_Wake_I2C()

```
bool BMP280_Wake_I2C (
            I2C_HandleTypeDef i2c_handle,
            uint8_t device_address )
```

Wake sensor over I2C - used when measuring in forced mode.

Parameters

i2c_handle	Desired MCU I2C peripheral for communication with sensor
device_address	I2C device address BMP280_DEVICE_ADDRESS_GND = 0x76 BMP280_DEVICE_ADDRESS_VDDIO = 0x77

Returns

Wake status
false == unsuccessful
true == successful

Wake sensor by writing MEASURE_MODE_FORCED bit to CTRL_MEAS register

5.2 Core/Inc/main.h File Reference

: Header for main.c file. This file contains the common defines of the application.

```
#include ¨stm32f1xx_hal.h¨
#include <stdbool.h>
#include <stdio.h>
```

Macros

- #define ON_BOARD_LED_2_Pin GPIO_PIN_13
- #define ON_BOARD_LED_2_GPIO_Port GPIOC
- #define ON_BOARD_LED_1_Pin GPIO_PIN_2
- #define ON_BOARD_LED_1_GPIO_Port GPIOB

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.
- int __io_putchar (int ch)

  putchar() override - redirect printf to USART2

5.2.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

5.2.2 Macro Definition Documentation

5.2.2.1 ON_BOARD_LED_1_GPIO_Port

```
#define ON_BOARD_LED_1_GPIO_Port GPIOB
```

5.2.2.2 ON_BOARD_LED_1_Pin

```
#define ON_BOARD_LED_1_Pin GPIO_PIN_2
```

5.2.2.3 ON_BOARD_LED_2_GPIO_Port

```
#define ON_BOARD_LED_2_GPIO_Port GPIOC
```

5.2.2.4 ON_BOARD_LED_2_Pin

```
#define ON_BOARD_LED_2_Pin GPIO_PIN_13
```

5.2.3 Function Documentation

5.2.3.1 __io_putchar()

```
int __io_putchar (
             int ch )
```

putchar() override - redirect printf to USART2

5.2.3.2 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

5.3 Core/Src/BMP280.c File Reference

BMP280 driver file.

```
#include ¨BMP280.h¨
```

Functions

- void BMP280_CalibrationConstantsRead_I2C (I2C_HandleTypeDef i2c_handle, uint8_t device_address)

  Read sensor calibration parameters over I2C.
- bool BMP280_Init_I2C (uint8_t osrs_t, uint8_t osrs_p, uint8_t acq_mode, uint8_t t_sb, uint8_t filter_tc, I2C↩ _HandleTypeDef i2c_handle, uint8_t device_address)

  Initialize sensor with chosen settings.
- bool BMP280_Wake_I2C (I2C_HandleTypeDef i2c_handle, uint8_t device_address)

  Wake sensor over I2C - used when measuring in forced mode.
- struct BMP280_Result BMP280_Measure_I2C (I2C_HandleTypeDef i2c_handle, uint8_t device_address)

  Measure temperature and pressure over I2C.

Variables

- int32_t t_fine

5.3.1 Detailed Description

BMP280 driver file.

Created on: Apr 5, 2024
Author: Piotr Jucha

5.3.2 Function Documentation

5.3.2.1 BMP280_CalibrationConstantsRead_I2C()

```
void BMP280_CalibrationConstantsRead_I2C (
            I2C_HandleTypeDef i2c_handle,
            uint8_t device_address )
```

Read sensor calibration parameters over I2C.

Read constants used for temperature and pressure calculations from sensor's memory

5.3.2.2 BMP280_Init_I2C()

```
bool BMP280_Init_I2C (
            uint8_t osrs_t,
            uint8_t osrs_p,
            uint8_t acq_mode,
            uint8_t t_sb,
            uint8_t filter_tc,
            I2C_HandleTypeDef i2c_handle,
            uint8_t device_address )
```

Initialize sensor with chosen settings.

Reset the sensor, check if device ID is valid, read calibration constants, write oversampling, acquisition mode, readout timing and filter data to the sensor

5.3.2.3 BMP280_Measure_I2C()

```
struct BMP280_Result BMP280_Measure_I2C (
            I2C_HandleTypeDef i2c_handle,
            uint8_t device_address )
```

Measure temperature and pressure over I2C.

Parameters

i2c_handle	Desired MCU I2C peripheral for communication with sensor
device_address	I2C device address BMP280_DEVICE_ADDRESS_GND = 0x76 BMP280_DEVICE_ADDRESS_VDDIO = 0x77

Returns

Measurement values

5.3.2.4 BMP280_Wake_I2C()

```
bool BMP280_Wake_I2C (
            I2C_HandleTypeDef i2c_handle,
            uint8_t device_address )
```

Wake sensor over I2C - used when measuring in forced mode.

Wake sensor by writing MEASURE_MODE_FORCED bit to CTRL_MEAS register

5.3.3 Variable Documentation

5.3.3.1 t_fine

```
int32_t t_fine
```

5.4 Core/Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "cmsis_os.h"
#include "gpio.h"
#include "i2c.h"
#include "usart.h"
#include "BMP280.h"
```

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- void MX_FREERTOS_Init (void)
- int main (void)

  The application entry point.
- int __io_putchar (int ch)

  putchar() override - redirect printf to USART2
- void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim)

  Period elapsed callback in non blocking mode.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

5.4.1 Detailed Description

: Main program body

Attention

5.4.2 Function Documentation

5.4.2.1 __io_putchar()

```
int __io_putchar (
            int ch )
```

putchar() override - redirect printf to USART2

5.4.2.2 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Return values

None	

5.4.2.3 HAL_TIM_PeriodElapsedCallback()

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

Period elapsed callback in non blocking mode.

Note

This function is called when TIM4 interrupt took place, inside HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment a global variable ¨uwTick¨ used as application time base.

Parameters

htim	: TIM handle

Return values

None	

5.4.2.4 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

5.4.2.5 MX_FREERTOS_Init()

```
void MX_FREERTOS_Init (
             void  )
```

5.4.2.6 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

5.5 README.md File Reference