task_2_doxygen

Emmanuel Benard and Kaleb Croft
Version 1

i

Mon May 11 2020

Table of Contents

Table of contents

Main Page

File **bme680.c**

Date

19 Jun 2018

Version

3.5.9

Module Index

Modules

Here is a list of all modules:

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Module Documentation

SENSOR API

Data Structures

- struct **bme680_field_data**
  Sensor field data structure.
- struct **bme680_calib_data**
  Structure to hold the Calibration data.
- struct **bme680_tph_sett**
  BME680 sensor settings structure which comprises of ODR, over-sampling and filter settings.
- struct **bme680_gas_sett**
  BME680 gas sensor which comprises of gas settings and status parameters.
- struct **bme680_dev**
  BME680 device structure.

Functions

- int8_t **bme680_init** (struct **bme680_dev** *dev)
  This API is the entry point. It reads the chip-id and calibration data from the sensor.

- int8_t **bme680_set_regs** (const uint8_t *reg_addr, const uint8_t *reg_data, uint8_t len, struct **bme680_dev** *dev)
  This API writes the given data to the register address of the sensor.

- int8_t **bme680_get_regs** (uint8_t reg_addr, uint8_t *reg_data, uint16_t len, struct **bme680_dev** *dev)
  This API reads the data from the given register address of the sensor.

- int8_t **bme680_soft_reset** (struct **bme680_dev** *dev)

This API performs the soft reset of the sensor.

- int8_t **bme680_set_sensor_mode** (struct **bme680_dev** *dev)
  This API is used to set the power mode of the sensor.

- int8_t **bme680_get_sensor_mode** (struct **bme680_dev** *dev)
  This API is used to get the power mode of the sensor.

- void **bme680_set_profile_dur** (uint16_t duration, struct **bme680_dev** *dev)
  This API is used to set the profile duration of the sensor.

- void **bme680_get_profile_dur** (uint16_t *duration, const struct **bme680_dev** *dev)
  This API is used to get the profile duration of the sensor.

- int8_t **bme680_get_sensor_data** (struct **bme680_field_data** *data, struct **bme680_dev** *dev)
  This API reads the pressure, temperature and humidity and gas data from the sensor, compensates the data and store it in the bme680_data structure instance passed by the user.

- int8_t **bme680_set_sensor_settings** (uint16_t desired_settings, struct **bme680_dev** *dev)
  This API is used to set the oversampling, filter and T,P,H, gas selection settings in the sensor.

- int8_t **bme680_get_sensor_settings** (uint16_t desired_settings, struct **bme680_dev** *dev)
  This API is used to get the oversampling, filter and T,P,H, gas selection settings in the sensor.

Common macros

- #define **INT8_C**(x) S8_C(x)
- #define **UINT8_C**(x) U8_C(x)
- #define **INT16_C**(x) S16_C(x)
- #define **UINT16_C**(x) U16_C(x)
- #define **INT32_C**(x) S32_C(x)
- #define **UINT32_C**(x) U32_C(x)
- #define **INT64_C**(x) S64_C(x)
- #define **UINT64_C**(x) U64_C(x)

C standard macros

- enum **bme680_intf** { **BME680_SPI_INTF**, **BME680_I2C_INTF** }
  Interface selection Enumerations.
- typedef int8_t(* **bme680_com_fptr_t**) (uint8_t dev_id, uint8_t reg_addr, uint8_t *data, uint16_t len)
- typedef void(* **bme680_delay_fptr_t**) (uint32_t period)
- #define **NULL** ((void *) 0)
- #define **BME680_POLL_PERIOD_MS** **UINT8_C**(10)
- #define **BME680_I2C_ADDR_PRIMARY** **UINT8_C**(0x76)
- #define **BME680_I2C_ADDR_SECONDARY** **UINT8_C**(0x77)
- #define **BME680_CHIP_ID** **UINT8_C**(0x61)
- #define **BME680_COEFF_SIZE** **UINT8_C**(41)
- #define **BME680_COEFF_ADDR1_LEN** **UINT8_C**(25)
- #define **BME680_COEFF_ADDR2_LEN** **UINT8_C**(16)
- #define **BME680_FIELD_LENGTH** **UINT8_C**(15)
- #define **BME680_FIELD_ADDR_OFFSET** **UINT8_C**(17)
- #define **BME680_SOFT_RESET_CMD** **UINT8_C**(0xb6)
- #define **BME680_OK** **INT8_C**(0)
- #define **BME680_E_NULL_PTR** **INT8_C**(-1)
- #define **BME680_E_COM_FAIL** **INT8_C**(-2)
- #define **BME680_E_DEV_NOT_FOUND** **INT8_C**(-3)
- #define **BME680_E_INVALID_LENGTH** **INT8_C**(-4)
- #define **BME680_W_DEFINE_PWR_MODE** **INT8_C**(1)
- #define **BME680_W_NO_NEW_DATA** **INT8_C**(2)
- #define **BME680_I_MIN_CORRECTION** **UINT8_C**(1)

- #define **BME680_I_MAX_CORRECTION UINT8_C**(2)
- #define **BME680_ADDR_RES_HEAT_VAL_ADDR UINT8_C**(0x00)
- #define **BME680_ADDR_RES_HEAT_RANGE_ADDR UINT8_C**(0x02)
- #define **BME680_ADDR_RANGE_SW_ERR_ADDR UINT8_C**(0x04)
- #define **BME680_ADDR_SENS_CONF_START UINT8_C**(0x5A)
- #define **BME680_ADDR_GAS_CONF_START UINT8_C**(0x64)
- #define **BME680_FIELD0_ADDR UINT8_C**(0x1d)
- #define **BME680_RES_HEAT0_ADDR UINT8_C**(0x5a)
- #define **BME680_GAS_WAIT0_ADDR UINT8_C**(0x64)
- #define **BME680_CONF_HEAT_CTRL_ADDR UINT8_C**(0x70)
- #define **BME680_CONF_ODR_RUN_GAS_NBC_ADDR UINT8_C**(0x71)
- #define **BME680_CONF_OS_H_ADDR UINT8_C**(0x72)
- #define **BME680_MEM_PAGE_ADDR UINT8_C**(0xf3)
- #define **BME680_CONF_T_P_MODE_ADDR UINT8_C**(0x74)
- #define **BME680_CONF_ODR_FILT_ADDR UINT8_C**(0x75)
- #define **BME680_COEFF_ADDR1 UINT8_C**(0x89)
- #define **BME680_COEFF_ADDR2 UINT8_C**(0xe1)
- #define **BME680_CHIP_ID_ADDR UINT8_C**(0xd0)
- #define **BME680_SOFT_RESET_ADDR UINT8_C**(0xe0)
- #define **BME680_ENABLE_HEATER UINT8_C**(0x00)
- #define **BME680_DISABLE_HEATER UINT8_C**(0x08)
- #define **BME680_DISABLE_GAS_MEAS UINT8_C**(0x00)
- #define **BME680_ENABLE_GAS_MEAS UINT8_C**(0x01)
- #define **BME680_OS_NONE UINT8_C**(0)
- #define **BME680_OS_1X UINT8_C**(1)
- #define **BME680_OS_2X UINT8_C**(2)
- #define **BME680_OS_4X UINT8_C**(3)
- #define **BME680_OS_8X UINT8_C**(4)
- #define **BME680_OS_16X UINT8_C**(5)
- #define **BME680_FILTER_SIZE_0 UINT8_C**(0)
- #define **BME680_FILTER_SIZE_1 UINT8_C**(1)
- #define **BME680_FILTER_SIZE_3 UINT8_C**(2)
- #define **BME680_FILTER_SIZE_7 UINT8_C**(3)

- #define **BME680_FILTER_SIZE_15 UINT8_C**(4)
- #define **BME680_FILTER_SIZE_31 UINT8_C**(5)
- #define **BME680_FILTER_SIZE_63 UINT8_C**(6)
- #define **BME680_FILTER_SIZE_127 UINT8_C**(7)
- #define **BME680_SLEEP_MODE UINT8_C**(0)
- #define **BME680_FORCED_MODE UINT8_C**(1)
- #define **BME680_RESET_PERIOD UINT32_C**(10)
- #define **BME680_MEM_PAGE0 UINT8_C**(0x10)
- #define **BME680_MEM_PAGE1 UINT8_C**(0x00)
- #define **BME680_HUM_REG_SHIFT_VAL UINT8_C**(4)
- #define **BME680_RUN_GAS_DISABLE UINT8_C**(0)
- #define **BME680_RUN_GAS_ENABLE UINT8_C**(1)
- #define **BME680_TMP_BUFFER_LENGTH UINT8_C**(40)
- #define **BME680_REG_BUFFER_LENGTH UINT8_C**(6)
- #define **BME680_FIELD_DATA_LENGTH UINT8_C**(3)
- #define **BME680_GAS_REG_BUF_LENGTH UINT8_C**(20)
- #define **BME680_OST_SEL UINT16_C**(1)
- #define **BME680_OSP_SEL UINT16_C**(2)
- #define **BME680_OSH_SEL UINT16_C**(4)
- #define **BME680_GAS_MEAS_SEL UINT16_C**(8)
- #define **BME680_FILTER_SEL UINT16_C**(16)
- #define **BME680_HCNTRL_SEL UINT16_C**(32)
- #define **BME680_RUN_GAS_SEL UINT16_C**(64)
- #define **BME680_NBCONV_SEL UINT16_C**(128)
- #define **BME680_GAS_SENSOR_SEL (BME680_GAS_MEAS_SEL** | **BME680_RUN_GAS_SEL** | **BME680_NBCONV_SEL)**
- #define **BME680_NBCONV_MIN UINT8_C**(0)
- #define **BME680_NBCONV_MAX UINT8_C**(10)
- #define **BME680_GAS_MEAS_MSK UINT8_C**(0x30)
- #define **BME680_NBCONV_MSK UINT8_C**(0X0F)
- #define **BME680_FILTER_MSK UINT8_C**(0X1C)
- #define **BME680_OST_MSK UINT8_C**(0XE0)
- #define **BME680_OSP_MSK UINT8_C**(0X1C)
- #define **BME680_OSH_MSK UINT8_C**(0X07)

- #define **BME680_HCTRL_MSK UINT8_C**(0x08)
- #define **BME680_RUN_GAS_MSK UINT8_C**(0x10)
- #define **BME680_MODE_MSK UINT8_C**(0x03)
- #define **BME680_RHRANGE_MSK UINT8_C**(0x30)
- #define **BME680_RSERROR_MSK UINT8_C**(0xf0)
- #define **BME680_NEW_DATA_MSK UINT8_C**(0x80)
- #define **BME680_GAS_INDEX_MSK UINT8_C**(0x0f)
- #define **BME680_GAS_RANGE_MSK UINT8_C**(0x0f)
- #define **BME680_GASM_VALID_MSK UINT8_C**(0x20)
- #define **BME680_HEAT_STAB_MSK UINT8_C**(0x10)
- #define **BME680_MEM_PAGE_MSK UINT8_C**(0x10)
- #define **BME680_SPI_RD_MSK UINT8_C**(0x80)
- #define **BME680_SPI_WR_MSK UINT8_C**(0x7f)
- #define **BME680_BIT_H1_DATA_MSK UINT8_C**(0x0F)
- #define **BME680_GAS_MEAS_POS UINT8_C**(4)
- #define **BME680_FILTER_POS UINT8_C**(2)
- #define **BME680_OST_POS UINT8_C**(5)
- #define **BME680_OSP_POS UINT8_C**(2)
- #define **BME680_RUN_GAS_POS UINT8_C**(4)
- #define **BME680_T2_LSB_REG** (1)
- #define **BME680_T2_MSB_REG** (2)
- #define **BME680_T3_REG** (3)
- #define **BME680_P1_LSB_REG** (5)
- #define **BME680_P1_MSB_REG** (6)
- #define **BME680_P2_LSB_REG** (7)
- #define **BME680_P2_MSB_REG** (8)
- #define **BME680_P3_REG** (9)
- #define **BME680_P4_LSB_REG** (11)
- #define **BME680_P4_MSB_REG** (12)
- #define **BME680_P5_LSB_REG** (13)
- #define **BME680_P5_MSB_REG** (14)
- #define **BME680_P7_REG** (15)
- #define **BME680_P6_REG** (16)

- #define **BME680_P8_LSB_REG** (19)
- #define **BME680_P8_MSB_REG** (20)
- #define **BME680_P9_LSB_REG** (21)
- #define **BME680_P9_MSB_REG** (22)
- #define **BME680_P10_REG** (23)
- #define **BME680_H2_MSB_REG** (25)
- #define **BME680_H2_LSB_REG** (26)
- #define **BME680_H1_LSB_REG** (26)
- #define **BME680_H1_MSB_REG** (27)
- #define **BME680_H3_REG** (28)
- #define **BME680_H4_REG** (29)
- #define **BME680_H5_REG** (30)
- #define **BME680_H6_REG** (31)
- #define **BME680_H7_REG** (32)
- #define **BME680_T1_LSB_REG** (33)
- #define **BME680_T1_MSB_REG** (34)
- #define **BME680_GH2_LSB_REG** (35)
- #define **BME680_GH2_MSB_REG** (36)
- #define **BME680_GH1_REG** (37)
- #define **BME680_GH3_REG** (38)
- #define **BME680_REG_FILTER_INDEX** **UINT8_C**(5)
- #define **BME680_REG_TEMP_INDEX** **UINT8_C**(4)
- #define **BME680_REG_PRES_INDEX** **UINT8_C**(4)
- #define **BME680_REG_HUM_INDEX** **UINT8_C**(2)
- #define **BME680_REG_NBCONV_INDEX** **UINT8_C**(1)
- #define **BME680_REG_RUN_GAS_INDEX** **UINT8_C**(1)
- #define **BME680_REG_HCTRL_INDEX** **UINT8_C**(0)
- #define **BME680_MAX_OVERFLOW_VAL** **INT32_C**(0x40000000)
- #define **BME680_CONCAT_BYTES**(msb, lsb) (((uint16_t)msb << 8) | (uint16_t)lsb)
- #define **BME680_SET_BITS**(reg_data, bitname, data)
- #define **BME680_GET_BITS**(reg_data, bitname)
- #define **BME680_SET_BITS_POS_0**(reg_data, bitname, data)
- #define **BME680_GET_BITS_POS_0**(reg_data, bitname) (reg_data & (bitname##_MSK))

Detailed Description

Macro Definition Documentation

#define BME680_ADDR_GAS_CONF_START UINT8_C(0x64)

Definition at line 153 of file bme680_defs.h.

#define BME680_ADDR_RANGE_SW_ERR_ADDR UINT8_C(0x04)

Definition at line 151 of file bme680_defs.h.

#define BME680_ADDR_RES_HEAT_RANGE_ADDR UINT8_C(0x02)

Definition at line 150 of file bme680_defs.h.

#define BME680_ADDR_RES_HEAT_VAL_ADDR UINT8_C(0x00)

Register map Other coefficient's address

Definition at line 149 of file bme680_defs.h.

#define BME680_ADDR_SENS_CONF_START UINT8_C(0x5A)

Definition at line 152 of file bme680_defs.h.

#define BME680_BIT_H1_DATA_MSK UINT8_C(0x0F)

Definition at line 265 of file bme680_defs.h.

#define BME680_CHIP_ID UINT8_C(0x61)

BME680 unique chip identifier

Definition at line 117 of file bme680_defs.h.

#define BME680_CHIP_ID_ADDR UINT8_C(0xd0)

Chip identifier

Definition at line 175 of file bme680_defs.h.

#define BME680_COEFF_ADDR1 UINT8_C(0x89)

Coefficient's address

Definition at line 171 of file bme680_defs.h.

#define BME680_COEFF_ADDR1_LEN UINT8_C(25)

Definition at line 121 of file bme680_defs.h.

#define BME680_COEFF_ADDR2 UINT8_C(0xe1)

Definition at line 172 of file bme680_defs.h.

#define BME680_COEFF_ADDR2_LEN UINT8_C(16)

Definition at line 122 of file bme680_defs.h.

#define BME680_COEFF_SIZE UINT8_C(41)

BME680 coefficients related defines

Definition at line 120 of file bme680_defs.h.

#define BME680_CONCAT_BYTES(msb, lsb) (((uint16_t)msb << 8) | (uint16_t)lsb)

Macro to combine two 8 bit data's to form a 16 bit data

Definition at line 328 of file bme680_defs.h.

#define BME680_CONF_HEAT_CTRL_ADDR UINT8_C(0x70)

Sensor configuration registers

Definition at line 163 of file bme680_defs.h.

#define BME680_CONF_ODR_FILT_ADDR UINT8_C(0x75)

Definition at line 168 of file bme680_defs.h.

#define BME680_CONF_ODR_RUN_GAS_NBC_ADDR UINT8_C(0x71)

Definition at line 164 of file bme680_defs.h.

#define BME680_CONF_OS_H_ADDR UINT8_C(0x72)

Definition at line 165 of file bme680_defs.h.

#define BME680_CONF_T_P_MODE_ADDR UINT8_C(0x74)

Definition at line 167 of file bme680_defs.h.

#define BME680_DISABLE_GAS_MEAS UINT8_C(0x00)

Gas measurement settings

Definition at line 185 of file bme680_defs.h.

#define BME680_DISABLE_HEATER UINT8_C(0x08)

Definition at line 182 of file bme680_defs.h.

#define BME680_E_COM_FAIL INT8_C(-2)

Definition at line 135 of file bme680_defs.h.

#define BME680_E_DEV_NOT_FOUND INT8_C(-3)

Definition at line 136 of file bme680_defs.h.

#define BME680_E_INVALID_LENGTH INT8_C(-4)

Definition at line 137 of file bme680_defs.h.

#define BME680_E_NULL_PTR INT8_C(-1)

Definition at line 134 of file bme680_defs.h.

#define BME680_ENABLE_GAS_MEAS UINT8_C(0x01)

Definition at line 186 of file bme680_defs.h.

#define BME680_ENABLE_HEATER UINT8_C(0x00)

Heater control settings

Definition at line 181 of file bme680_defs.h.

#define BME680_FIELD0_ADDR UINT8_C(0x1d)

Field settings

Definition at line 156 of file bme680_defs.h.

#define BME680_FIELD_ADDR_OFFSET UINT8_C(17)

Definition at line 126 of file bme680_defs.h.

#define BME680_FIELD_DATA_LENGTH UINT8_C(3)

Definition at line 227 of file bme680_defs.h.

#define BME680_FIELD_LENGTH UINT8_C(15)

BME680 field_x related defines

Definition at line 125 of file bme680_defs.h.

#define BME680_FILTER_MSK UINT8_C(0X1C)

Definition at line 248 of file bme680_defs.h.

#define BME680_FILTER_POS UINT8_C(2)

Definition at line 269 of file bme680_defs.h.

#define BME680_FILTER_SEL UINT16_C(16)

Definition at line 235 of file bme680_defs.h.

#define BME680_FILTER_SIZE_0 UINT8_C(0)

IIR filter settings

Definition at line 197 of file bme680_defs.h.

#define BME680_FILTER_SIZE_1 UINT8_C(1)

Definition at line 198 of file bme680_defs.h.

#define BME680_FILTER_SIZE_127 UINT8_C(7)

Definition at line 204 of file bme680_defs.h.

#define BME680_FILTER_SIZE_15 UINT8_C(4)

Definition at line 201 of file bme680_defs.h.

#define BME680_FILTER_SIZE_3 UINT8_C(2)

Definition at line 199 of file bme680_defs.h.

#define BME680_FILTER_SIZE_31 UINT8_C(5)

Definition at line 202 of file bme680_defs.h.

#define BME680_FILTER_SIZE_63 UINT8_C(6)

Definition at line 203 of file bme680_defs.h.

#define BME680_FILTER_SIZE_7 UINT8_C(3)

Definition at line 200 of file bme680_defs.h.

#define BME680_FORCED_MODE UINT8_C(1)

Definition at line 208 of file bme680_defs.h.

#define BME680_GAS_INDEX_MSK UINT8_C(0x0f)

Definition at line 258 of file bme680_defs.h.

#define BME680_GAS_MEAS_MSK UINT8_C(0x30)

Mask definitions

Definition at line 246 of file bme680_defs.h.

#define BME680_GAS_MEAS_POS UINT8_C(4)

Bit position definitions for sensor settings

Definition at line 268 of file bme680_defs.h.

#define BME680_GAS_MEAS_SEL UINT16_C(8)

Definition at line 234 of file bme680_defs.h.

#define BME680_GAS_RANGE_MSK UINT8_C(0x0f)

Definition at line 259 of file bme680_defs.h.

#define BME680_GAS_REG_BUF_LENGTH UINT8_C(20)

Definition at line 228 of file bme680_defs.h.

#define BME680_GAS_SENSOR_SEL (BME680_GAS_MEAS_SEL | BME680_RUN_GAS_SEL | BME680_NBCONV_SEL)

Definition at line 239 of file bme680_defs.h.

#define BME680_GAS_WAIT0_ADDR UINT8_C(0x64)

Definition at line 160 of file bme680_defs.h.

#define BME680_GASM_VALID_MSK UINT8_C(0x20)

Definition at line 260 of file bme680_defs.h.

#define BME680_GET_BITS(reg_data, bitname)

```
Value:    ((reg_data & (bitname##_MSK)) >> \
    (bitname##_POS))
```

Definition at line 334 of file bme680_defs.h.

#define BME680_GET_BITS_POS_0(reg_data, bitname) (reg_data & (bitname##_MSK))

Definition at line 341 of file bme680_defs.h.

#define BME680_GH1_REG (37)

Definition at line 307 of file bme680_defs.h.

#define BME680_GH2_LSB_REG (35)

Definition at line 305 of file bme680_defs.h.

#define BME680_GH2_MSB_REG (36)

Definition at line 306 of file bme680_defs.h.

#define BME680_GH3_REG (38)

Definition at line 308 of file bme680_defs.h.

#define BME680_H1_LSB_REG (26)

Definition at line 296 of file bme680_defs.h.

#define BME680_H1_MSB_REG (27)

Definition at line 297 of file bme680_defs.h.

#define BME680_H2_LSB_REG (26)

Definition at line 295 of file bme680_defs.h.

#define BME680_H2_MSB_REG (25)

Definition at line 294 of file bme680_defs.h.

#define BME680_H3_REG (28)

Definition at line 298 of file bme680_defs.h.

#define BME680_H4_REG (29)

Definition at line 299 of file bme680_defs.h.

#define BME680_H5_REG (30)

Definition at line 300 of file bme680_defs.h.

#define BME680_H6_REG (31)

Definition at line 301 of file bme680_defs.h.

#define BME680_H7_REG (32)

Definition at line 302 of file bme680_defs.h.

#define BME680_HCNTRL_SEL UINT16_C(32)

Definition at line 236 of file bme680_defs.h.

#define BME680_HCTRL_MSK UINT8_C(0x08)

Definition at line 252 of file bme680_defs.h.

#define BME680_HEAT_STAB_MSK UINT8_C(0x10)

Definition at line 261 of file bme680_defs.h.

#define BME680_HUM_REG_SHIFT_VAL UINT8_C(4)

Ambient humidity shift value for compensation

Definition at line 218 of file bme680_defs.h.

#define BME680_I2C_ADDR_PRIMARY UINT8_C(0x76)

BME680 I2C addresses

Definition at line 113 of file bme680_defs.h.

#define BME680_I2C_ADDR_SECONDARY UINT8_C(0x77)

Definition at line 114 of file bme680_defs.h.

#define BME680_I_MAX_CORRECTION UINT8_C(2)

Definition at line 145 of file bme680_defs.h.

#define BME680_I_MIN_CORRECTION UINT8_C(1)

Definition at line 144 of file bme680_defs.h.

#define BME680_MAX_OVERFLOW_VAL INT32_C(0x40000000)

BME680 pressure calculation macros

This max value is used to provide precedence to multiplication or division in pressure compensation equation to achieve least loss of precision and avoiding overflows. i.e Comparing value, BME680_MAX_OVERFLOW_VAL = **INT32_C(1 << 30)**

Definition at line 325 of file bme680_defs.h.

#define BME680_MEM_PAGE0 UINT8_C(0x10)

SPI memory page settings

Definition at line 214 of file bme680_defs.h.

#define BME680_MEM_PAGE1 UINT8_C(0x00)

Definition at line 215 of file bme680_defs.h.

#define BME680_MEM_PAGE_ADDR UINT8_C(0xf3)

Definition at line 166 of file bme680_defs.h.

#define BME680_MEM_PAGE_MSK UINT8_C(0x10)

Definition at line 262 of file bme680_defs.h.

#define BME680_MODE_MSK UINT8_C(0x03)

Definition at line 254 of file bme680_defs.h.

#define BME680_NBCONV_MAX UINT8_C(10)

Definition at line 243 of file bme680_defs.h.

#define BME680_NBCONV_MIN UINT8_C(0)

Number of conversion settings

Definition at line 242 of file bme680_defs.h.

#define BME680_NBCONV_MSK UINT8_C(0X0F)

Definition at line 247 of file bme680_defs.h.

#define BME680_NBCONV_SEL UINT16_C(128)

Definition at line 238 of file bme680_defs.h.

#define BME680_NEW_DATA_MSK UINT8_C(0x80)

Definition at line 257 of file bme680_defs.h.

#define BME680_OK INT8_C(0)

Error code definitions

Definition at line 132 of file bme680_defs.h.

#define BME680_OS_16X UINT8_C(5)

Definition at line 194 of file bme680_defs.h.

#define BME680_OS_1X UINT8_C(1)

Definition at line 190 of file bme680_defs.h.

#define BME680_OS_2X UINT8_C(2)

Definition at line 191 of file bme680_defs.h.

#define BME680_OS_4X UINT8_C(3)

Definition at line 192 of file bme680_defs.h.

#define BME680_OS_8X UINT8_C(4)

Definition at line 193 of file bme680_defs.h.

#define BME680_OS_NONE UINT8_C(0)

Over-sampling settings

Definition at line 189 of file bme680_defs.h.

#define BME680_OSH_MSK UINT8_C(0X07)

Definition at line 251 of file bme680_defs.h.

#define BME680_OSH_SEL UINT16_C(4)

Definition at line 233 of file bme680_defs.h.

#define BME680_OSP_MSK UINT8_C(0X1C)

Definition at line 250 of file bme680_defs.h.

#define BME680_OSP_POS UINT8_C(2)

Definition at line 271 of file bme680_defs.h.

#define BME680_OSP_SEL UINT16_C(2)

Definition at line 232 of file bme680_defs.h.

#define BME680_OST_MSK UINT8_C(0XE0)

Definition at line 249 of file bme680_defs.h.

#define BME680_OST_POS UINT8_C(5)

Definition at line 270 of file bme680_defs.h.

#define BME680_OST_SEL UINT16_C(1)

Settings selector

Definition at line 231 of file bme680_defs.h.

#define BME680_P10_REG (23)

Definition at line 293 of file bme680_defs.h.

#define BME680_P1_LSB_REG (5)

Definition at line 278 of file bme680_defs.h.

#define BME680_P1_MSB_REG (6)

Definition at line 279 of file bme680_defs.h.

#define BME680_P2_LSB_REG (7)

Definition at line 280 of file bme680_defs.h.

#define BME680_P2_MSB_REG (8)

Definition at line 281 of file bme680_defs.h.

#define BME680_P3_REG (9)

Definition at line 282 of file bme680_defs.h.

#define BME680_P4_LSB_REG (11)

Definition at line 283 of file bme680_defs.h.

#define BME680_P4_MSB_REG (12)

Definition at line 284 of file bme680_defs.h.

#define BME680_P5_LSB_REG (13)

Definition at line 285 of file bme680_defs.h.

#define BME680_P5_MSB_REG (14)

Definition at line 286 of file bme680_defs.h.

#define BME680_P6_REG (16)

Definition at line 288 of file bme680_defs.h.

#define BME680_P7_REG (15)

Definition at line 287 of file bme680_defs.h.

#define BME680_P8_LSB_REG (19)

Definition at line 289 of file bme680_defs.h.

#define BME680_P8_MSB_REG (20)

Definition at line 290 of file bme680_defs.h.

#define BME680_P9_LSB_REG (21)

Definition at line 291 of file bme680_defs.h.

#define BME680_P9_MSB_REG (22)

Definition at line 292 of file bme680_defs.h.

#define BME680_POLL_PERIOD_MS UINT8_C(10)

BME680 configuration macros Enable or un-comment the macro to provide floating point data output BME680 General config

Definition at line 110 of file bme680_defs.h.

#define BME680_REG_BUFFER_LENGTH UINT8_C(6)

Definition at line 226 of file bme680_defs.h.

#define BME680_REG_FILTER_INDEX UINT8_C(5)

BME680 register buffer index settings

Definition at line 311 of file bme680_defs.h.

#define BME680_REG_HCTRL_INDEX UINT8_C(0)

Definition at line 317 of file bme680_defs.h.

#define BME680_REG_HUM_INDEX UINT8_C(2)

Definition at line 314 of file bme680_defs.h.

#define BME680_REG_NBCONV_INDEX UINT8_C(1)

Definition at line 315 of file bme680_defs.h.

#define BME680_REG_PRES_INDEX UINT8_C(4)

Definition at line 313 of file bme680_defs.h.

#define BME680_REG_RUN_GAS_INDEX UINT8_C(1)

Definition at line 316 of file bme680_defs.h.

#define BME680_REG_TEMP_INDEX UINT8_C(4)

Definition at line 312 of file bme680_defs.h.

#define BME680_RES_HEAT0_ADDR UINT8_C(0x5a)

Heater settings

Definition at line 159 of file bme680_defs.h.

#define BME680_RESET_PERIOD UINT32_C(10)

Delay related macro declaration

Definition at line 211 of file bme680_defs.h.

#define BME680_RHRANGE_MSK UINT8_C(0x30)

Definition at line 255 of file bme680_defs.h.

#define BME680_RSERROR_MSK UINT8_C(0xf0)

Definition at line 256 of file bme680_defs.h.

#define BME680_RUN_GAS_DISABLE UINT8_C(0)

Run gas enable and disable settings

Definition at line 221 of file bme680_defs.h.

#define BME680_RUN_GAS_ENABLE UINT8_C(1)

Definition at line 222 of file bme680_defs.h.

#define BME680_RUN_GAS_MSK UINT8_C(0x10)

Definition at line 253 of file bme680_defs.h.

#define BME680_RUN_GAS_POS UINT8_C(4)

Definition at line 272 of file bme680_defs.h.

#define BME680_RUN_GAS_SEL UINT16_C(64)

Definition at line 237 of file bme680_defs.h.

#define BME680_SET_BITS(reg_data, bitname, data)

```
Value:        ((reg_data & ~(bitname##_MSK)) | \
       ((data << bitname##_POS) & bitname##_MSK))
```

Macro to SET and GET BITS of a register

Definition at line 331 of file bme680_defs.h.

#define BME680_SET_BITS_POS_0(reg_data, bitname, data)

```
Value:                ((reg_data & ~(bitname##_MSK)) | \
               (data & bitname##_MSK))
```

Macro variant to handle the bitname position if it is zero

Definition at line 338 of file bme680_defs.h.

#define BME680_SLEEP_MODE UINT8_C(0)

Power mode settings

Definition at line 207 of file bme680_defs.h.

#define BME680_SOFT_RESET_ADDR UINT8_C(0xe0)

Soft reset register

Definition at line 178 of file bme680_defs.h.

#define BME680_SOFT_RESET_CMD UINT8_C(0xb6)

Soft reset command

Definition at line 129 of file bme680_defs.h.

#define BME680_SPI_RD_MSK UINT8_C(0x80)

Definition at line 263 of file bme680_defs.h.

#define BME680_SPI_WR_MSK UINT8_C(0x7f)

Definition at line 264 of file bme680_defs.h.

#define BME680_T1_LSB_REG (33)

Definition at line 303 of file bme680_defs.h.

#define BME680_T1_MSB_REG (34)

Definition at line 304 of file bme680_defs.h.

#define BME680_T2_LSB_REG (1)

Array Index to Field data mapping for Calibration Data

Definition at line 275 of file bme680_defs.h.

#define BME680_T2_MSB_REG (2)

Definition at line 276 of file bme680_defs.h.

#define BME680_T3_REG (3)

Definition at line 277 of file bme680_defs.h.

#define BME680_TMP_BUFFER_LENGTH UINT8_C(40)

Buffer length macro declaration

Definition at line 225 of file bme680_defs.h.

#define BME680_W_DEFINE_PWR_MODE INT8_C(1)

Definition at line 140 of file bme680_defs.h.

#define BME680_W_NO_NEW_DATA INT8_C(2)

Definition at line 141 of file bme680_defs.h.

#define INT16_C(x) S16_C(x)

Definition at line 78 of file bme680_defs.h.

#define INT32_C(x) S32_C(x)

Definition at line 83 of file bme680_defs.h.

#define INT64_C(x) S64_C(x)

Definition at line 88 of file bme680_defs.h.

#define INT8_C(x) S8_C(x)

Definition at line 73 of file bme680_defs.h.

#define NULL ((void *) 0)

Definition at line 99 of file bme680_defs.h.

#define UINT16_C(x) U16_C(x)

Definition at line 79 of file bme680_defs.h.

#define UINT32_C(x) U32_C(x)

Definition at line 84 of file bme680_defs.h.

#define UINT64_C(x) U64_C(x)

Definition at line 89 of file bme680_defs.h.

#define UINT8_C(x) U8_C(x)

Definition at line 74 of file bme680_defs.h.

Typedef Documentation

typedef int8_t(* bme680_com_fptr_t) (uint8_t dev_id, uint8_t reg_addr, uint8_t *data, uint16_t len)

Type definitions

Generic communication function pointer

Parameters

in	*dev_id*	Place holder to store the id of the device structure Can be used to store the index of the Chip select or I2C address of the device.
in	*reg_addr*	Used to select the register the where data needs to be read from or written to.
	[in/out]	reg_data: Data array to read/write
in	*len*	Length of the data array

Definition at line 354 of file bme680_defs.h.

typedef void(* bme680_delay_fptr_t) (uint32_t period)

Delay function pointer

Parameters

in	*period*	Time period in milliseconds

Definition at line 360 of file bme680_defs.h.

Enumeration Type Documentation

enum bme680_intf

Interface selection Enumerations.

Enumerator:

BME680_SPI_INTF	SPI interface
BME680_I2C_INTF	I2C interface

Definition at line 365 of file bme680_defs.h.

Function Documentation

void bme680_get_profile_dur (uint16_t * *duration*, const struct bme680_dev * *dev*)

This API is used to get the profile duration of the sensor.

Parameters

in	*dev*	: Structure instance of **bme680_dev**.
in	*duration*	: Duration of the measurement in ms.

Returns

Nothing

Definition at line 672 of file bme680.c.

int8_t bme680_get_regs (uint8_t *reg_addr*, uint8_t * *reg_data*, uint16_t *len*, struct bme680_dev * *dev*)

This API reads the data from the given register address of the sensor.

Parameters

in	*reg_addr*	: Register address from where the data to be read
out	*reg_data*	: Pointer to data buffer to store the read data.
in	*len*	: No of bytes of data to be read.
in	*dev*	: Structure instance of **bme680_dev**.

Returns

Result of API execution status

Return values

zero	-> Success / +ve value -> Warning / -ve value -> Error

Definition at line 315 of file bme680.c.

int8_t bme680_get_sensor_data (struct bme680_field_data * *data*, struct bme680_dev * *dev*)

This API reads the pressure, temperature and humidity and gas data from the sensor, compensates the data and store it in the bme680_data structure instance passed by the user.

Parameters

out	*data*	Structure instance to hold the data.
in	*dev*	: Structure instance of **bme680_dev**.

Returns

Result of API execution status

Return values

zero	-> Success / +ve value -> Warning / -ve value -> Error

Definition at line 705 of file bme680.c.

int8_t bme680_get_sensor_mode (struct bme680_dev * *dev*)

This API is used to get the power mode of the sensor.

Parameters

in	*dev*	: Structure instance of **bme680_dev**

Note

: **bme680_dev.power_mode** structure variable hold the power mode.

value	mode
0x00	BME680_SLEEP_MODE
0x01	BME680_FORCED_MODE

Returns

Result of API execution status

Return values

zero	-> Success / +ve value -> Warning / -ve value -> Error

Definition at line 628 of file bme680.c.

int8_t bme680_get_sensor_settings (uint16_t *desired_settings*, struct bme680_dev * *dev*)

This API is used to get the oversampling, filter and T,P,H, gas selection settings in the sensor.

Parameters

in	*dev*	: Structure instance of **bme680_dev**.
in	*desired_settings*	: Variable used to select the settings which are to be get from the sensor.

Returns

Result of API execution status

Return values

zero	-> Success / +ve value -> Warning / -ve value -> Error.

Definition at line 537 of file bme680.c.

int8_t bme680_init (struct bme680_dev * *dev*)

This API is the entry point. It reads the chip-id and calibration data from the sensor.

CPP guard

Parameters

in,out	*dev*	: Structure instance of **bme680_dev**

Returns

Result of API execution status

Return values

zero	-> Success / +ve value -> Warning / -ve value -> Error

Definition at line 287 of file bme680.c.

void bme680_set_profile_dur (uint16_t *duration*, struct bme680_dev * *dev*)

This API is used to set the profile duration of the sensor.

Parameters

in	*dev*	: Structure instance of **bme680_dev**.
in	*duration*	: Duration of the measurement in ms.

Returns

Nothing

Definition at line 647 of file bme680.c.

int8_t bme680_set_regs (const uint8_t * *reg_addr*, const uint8_t * *reg_data*, uint8_t *len*, struct bme680_dev * *dev*)

This API writes the given data to the register address of the sensor.

Parameters

in	*reg_addr*	: Register address from where the data to be written.
in	*reg_data*	: Pointer to data buffer which is to be written in the sensor.
in	*len*	: No of bytes of data to write..
in	*dev*	: Structure instance of **bme680_dev**.

Returns

Result of API execution status

Return values

zero	-> Success / +ve value -> Warning / -ve value -> Error

Definition at line 340 of file bme680.c.

int8_t bme680_set_sensor_mode (struct bme680_dev * *dev*)

This API is used to set the power mode of the sensor.

Parameters

in	*dev*	: Structure instance of **bme680_dev**

Note

: Pass the value to **bme680_dev.power_mode** structure variable.

value	mode
0x00	BME680_SLEEP_MODE
0x01	BME680_FORCED_MODE

-
- **Returns**

  Result of API execution status

- **Return values**

zero	-> Success / +ve value -> Warning / -ve value -> Error

Definition at line 589 of file bme680.c.

int8_t bme680_set_sensor_settings (uint16_t *desired_settings*, struct bme680_dev * *dev*)

This API is used to set the oversampling, filter and T,P,H, gas selection settings in the sensor.

Parameters

in	*dev*	: Structure instance of **bme680_dev**.
in	*desired_settings*	: Variable used to select the settings which are to be set in the sensor. Macros | Functionality ---|--- BME680_OST_SEL | To set temperature oversampling. BME680_OSP_SEL | To set pressure oversampling. BME680_OSH_SEL | To set humidity oversampling. BME680_GAS_MEAS_SEL | To set gas measurement setting. BME680_FILTER_SEL | To set filter setting. BME680_HCNTRL_SEL | To set humidity control setting. BME680_RUN_GAS_SEL | To set run gas setting. BME680_NBCONV_SEL | To set NB conversion setting. BME680_GAS_SENSOR_SEL | To set all gas sensor related settings

Note

: Below are the macros to be used by the user for selecting the desired settings. User can do OR operation of these macros for configuring multiple settings.

Returns

Result of API execution status

Return values

zero	-> Success / +ve value -> Warning / -ve value -> Error.

Definition at line 413 of file bme680.c.

int8_t bme680_soft_reset (struct bme680_dev * *dev*)

This API performs the soft reset of the sensor.

Parameters

in	*dev*	: Structure instance of **bme680_dev**.

Returns

Result of API execution status

Return values

zero	-> Success / +ve value -> Warning / -ve value -> Error.

Definition at line 379 of file bme680.c.

Data Structure Documentation

bme680_calib_data Struct Reference

Structure to hold the Calibration data.

`#include <bme680_defs.h>`

Data Fields

- uint16_t **par_h1**
- uint16_t **par_h2**
- int8_t **par_h3**
- int8_t **par_h4**
- int8_t **par_h5**
- uint8_t **par_h6**
- int8_t **par_h7**
- int8_t **par_gh1**
- int16_t **par_gh2**
- int8_t **par_gh3**
- uint16_t **par_t1**
- int16_t **par_t2**
- int8_t **par_t3**
- uint16_t **par_p1**
- int16_t **par_p2**
- int8_t **par_p3**
- int16_t **par_p4**
- int16_t **par_p5**
- int8_t **par_p6**
- int8_t **par_p7**
- int16_t **par_p8**
- int16_t **par_p9**

- uint8_t **par_p10**
- int32_t **t_fine**
- uint8_t **res_heat_range**
- int8_t **res_heat_val**
- int8_t **range_sw_err**

Detailed Description

Structure to hold the Calibration data.

Definition at line 410 of file bme680_defs.h.

Field Documentation

int8_t par_gh1

Variable to store calibrated gas data

Definition at line 426 of file bme680_defs.h.

int16_t par_gh2

Variable to store calibrated gas data

Definition at line 428 of file bme680_defs.h.

int8_t par_gh3

Variable to store calibrated gas data

Definition at line 430 of file bme680_defs.h.

uint16_t par_h1

Variable to store calibrated humidity data

Definition at line 412 of file bme680_defs.h.

uint16_t par_h2

Variable to store calibrated humidity data

Definition at line 414 of file bme680_defs.h.

int8_t par_h3

Variable to store calibrated humidity data

Definition at line 416 of file bme680_defs.h.

int8_t par_h4

Variable to store calibrated humidity data

Definition at line 418 of file bme680_defs.h.

int8_t par_h5

Variable to store calibrated humidity data

Definition at line 420 of file bme680_defs.h.

uint8_t par_h6

Variable to store calibrated humidity data

Definition at line 422 of file bme680_defs.h.

int8_t par_h7

Variable to store calibrated humidity data

Definition at line 424 of file bme680_defs.h.

uint16_t par_p1

Variable to store calibrated pressure data

Definition at line 438 of file bme680_defs.h.

uint8_t par_p10

Variable to store calibrated pressure data

Definition at line 456 of file bme680_defs.h.

int16_t par_p2

Variable to store calibrated pressure data

Definition at line 440 of file bme680_defs.h.

int8_t par_p3

Variable to store calibrated pressure data

Definition at line 442 of file bme680_defs.h.

int16_t par_p4

Variable to store calibrated pressure data

Definition at line 444 of file bme680_defs.h.

int16_t par_p5

Variable to store calibrated pressure data

Definition at line 446 of file bme680_defs.h.

int8_t par_p6

Variable to store calibrated pressure data

Definition at line 448 of file bme680_defs.h.

int8_t par_p7

Variable to store calibrated pressure data

Definition at line 450 of file bme680_defs.h.

int16_t par_p8

Variable to store calibrated pressure data

Definition at line 452 of file bme680_defs.h.

int16_t par_p9

Variable to store calibrated pressure data

Definition at line 454 of file bme680_defs.h.

uint16_t par_t1

Variable to store calibrated temperature data

Definition at line 432 of file bme680_defs.h.

int16_t par_t2

Variable to store calibrated temperature data

Definition at line 434 of file bme680_defs.h.

int8_t par_t3

Variable to store calibrated temperature data

Definition at line 436 of file bme680_defs.h.

int8_t range_sw_err

Variable to store error range

Definition at line 470 of file bme680_defs.h.

uint8_t res_heat_range

Variable to store heater resistance range

Definition at line 466 of file bme680_defs.h.

int8_t res_heat_val

Variable to store heater resistance value

Definition at line 468 of file bme680_defs.h.

int32_t t_fine

Variable to store t_fine size

Definition at line 460 of file bme680_defs.h.

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

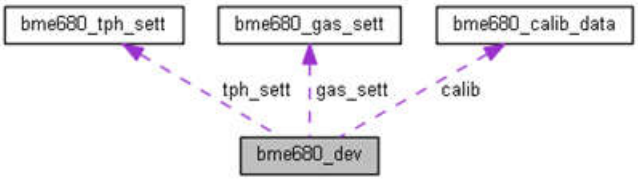
- F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/**bme680_defs.h**

bme680_dev Struct Reference

BME680 device structure.

`#include <bme680_defs.h>`

Collaboration diagram for bme680_dev:



Data Fields

- uint8_t **chip_id**
- uint8_t **dev_id**
- enum **bme680_intf intf**
- uint8_t **mem_page**
- int8_t **amb_temp**
- struct **bme680_calib_data calib**
- struct **bme680_tph_sett tph_sett**
- struct **bme680_gas_sett gas_sett**
- uint8_t **power_mode**
- uint8_t **new_fields**
- uint8_t **info_msg**
- **bme680_com_fptr_t read**
- **bme680_com_fptr_t write**
- **bme680_delay_fptr_t delay_ms**
- int8_t **com_rslt**

Detailed Description

BME680 device structure.

Definition at line 508 of file bme680_defs.h.

Field Documentation

int8_t amb_temp

Ambient temperature in Degree C

Definition at line 518 of file bme680_defs.h.

struct bme680_calib_data calib

Sensor calibration data

Definition at line 520 of file bme680_defs.h.

uint8_t chip_id

Chip Id

Definition at line 510 of file bme680_defs.h.

int8_t com_rslt

Communication function result

Definition at line 538 of file bme680_defs.h.

bme680_delay_fptr_t delay_ms

delay function pointer

Definition at line 536 of file bme680_defs.h.

uint8_t dev_id

Device Id

Definition at line 512 of file bme680_defs.h.

struct bme680_gas_sett gas_sett

Gas Sensor settings

Definition at line 524 of file bme680_defs.h.

uint8_t info_msg

Store the info messages

Definition at line 530 of file bme680_defs.h.

enum bme680_intf intf

SPI/I2C interface

Definition at line 514 of file bme680_defs.h.

uint8_t mem_page

Memory page used

Definition at line 516 of file bme680_defs.h.

uint8_t new_fields

New sensor fields

Definition at line 528 of file bme680_defs.h.

uint8_t power_mode

Sensor power modes

Definition at line 526 of file bme680_defs.h.

bme680_com_fptr_t read

Bus read function pointer

Definition at line 532 of file bme680_defs.h.

struct bme680_tph_sett tph_sett

Sensor settings

Definition at line 522 of file bme680_defs.h.

bme680_com_fptr_t write

Bus write function pointer

Definition at line 534 of file bme680_defs.h.

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

- F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/**bme680_defs.h**

bme680_field_data Struct Reference

Sensor field data structure.

`#include <bme680_defs.h>`

Data Fields

- uint8_t **status**
- uint8_t **gas_index**
- uint8_t **meas_index**
- int16_t **temperature**
- uint32_t **pressure**
- uint32_t **humidity**
- uint32_t **gas_resistance**

Detailed Description

Sensor field data structure.

Definition at line 376 of file bme680_defs.h.

Field Documentation

uint8_t gas_index

The index of the heater profile used

Definition at line 380 of file bme680_defs.h.

uint32_t gas_resistance

Gas resistance in Ohms

Definition at line 392 of file bme680_defs.h.

uint32_t humidity

Humidity in % relative humidity x1000

Definition at line 390 of file bme680_defs.h.

uint8_t meas_index

Measurement index to track order

Definition at line 382 of file bme680_defs.h.

uint32_t pressure

Pressure in Pascal

Definition at line 388 of file bme680_defs.h.

uint8_t status

Contains new_data, gasm_valid & heat_stab

Definition at line 378 of file bme680_defs.h.

int16_t temperature

Temperature in degree celsius x100

Definition at line 386 of file bme680_defs.h.

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

- F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/**bme680_defs.h**

bme680_gas_sett Struct Reference

BME680 gas sensor which comprises of gas settings and status parameters.

`#include <bme680_defs.h>`

Data Fields

- uint8_t **nb_conv**
- uint8_t **heatr_ctrl**
- uint8_t **run_gas**
- uint16_t **heatr_temp**
- uint16_t **heatr_dur**

Detailed Description

BME680 gas sensor which comprises of gas settings and status parameters.

Definition at line 492 of file bme680_defs.h.

Field Documentation

uint8_t heatr_ctrl

Variable to store heater control

Definition at line 496 of file bme680_defs.h.

uint16_t heatr_dur

Duration profile value

Definition at line 502 of file bme680_defs.h.

uint16_t heatr_temp

Heater temperature value

Definition at line 500 of file bme680_defs.h.

uint8_t nb_conv

Variable to store nb conversion

Definition at line 494 of file bme680_defs.h.

uint8_t run_gas

Run gas enable value

Definition at line 498 of file bme680_defs.h.

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

- F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/**bme680_defs.h**

bme680_tph_sett Struct Reference

BME680 sensor settings structure which comprises of ODR, over-sampling and filter settings.
`#include <bme680_defs.h>`

Data Fields

- uint8_t **os_hum**
- uint8_t **os_temp**
- uint8_t **os_pres**
- uint8_t **filter**

Detailed Description

BME680 sensor settings structure which comprises of ODR, over-sampling and filter settings.

Definition at line 477 of file bme680_defs.h.

Field Documentation

uint8_t filter

Filter coefficient

Definition at line 485 of file bme680_defs.h.

uint8_t os_hum

Humidity oversampling

Definition at line 479 of file bme680_defs.h.

uint8_t os_pres

Pressure oversampling

Definition at line 483 of file bme680_defs.h.

uint8_t os_temp

Temperature oversampling

Definition at line 481 of file bme680_defs.h.

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

- F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/**bme680_defs.h**

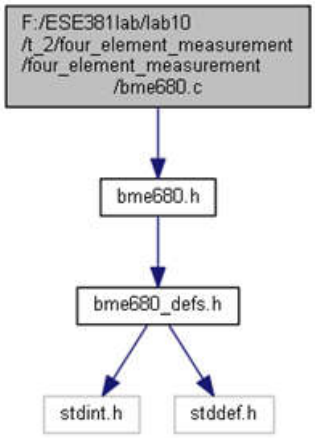
File Documentation

F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/bme680.c File Reference

Sensor driver for BME680 sensor.

```
#include "bme680.h"
```

Include dependency graph for bme680.c:



Functions

- int8_t **bme680_init** (struct **bme680_dev** *dev)
  This API is the entry point. It reads the chip-id and calibration data from the sensor.

- int8_t **bme680_get_regs** (uint8_t reg_addr, uint8_t *reg_data, uint16_t len, struct **bme680_dev** *dev)
  This API reads the data from the given register address of the sensor.

- int8_t **bme680_set_regs** (const uint8_t *reg_addr, const uint8_t *reg_data, uint8_t len, struct **bme680_dev** *dev)
  This API writes the given data to the register address of the sensor.

- int8_t **bme680_soft_reset** (struct **bme680_dev** *dev)
  This API performs the soft reset of the sensor.

- int8_t **bme680_set_sensor_settings** (uint16_t desired_settings, struct **bme680_dev** *dev)
  This API is used to set the oversampling, filter and T,P,H, gas selection settings in the sensor.

- int8_t **bme680_get_sensor_settings** (uint16_t desired_settings, struct **bme680_dev** *dev)
  This API is used to get the oversampling, filter and T,P,H, gas selection settings in the sensor.

- int8_t **bme680_set_sensor_mode** (struct **bme680_dev** *dev)
  This API is used to set the power mode of the sensor.

- int8_t **bme680_get_sensor_mode** (struct **bme680_dev** *dev)
  This API is used to get the power mode of the sensor.

- void **bme680_set_profile_dur** (uint16_t duration, struct **bme680_dev** *dev)
  This API is used to set the profile duration of the sensor.

- void **bme680_get_profile_dur** (uint16_t *duration, const struct **bme680_dev** *dev)
  This API is used to get the profile duration of the sensor.

- int8_t **bme680_get_sensor_data** (struct **bme680_field_data** *data, struct **bme680_dev** *dev)
  This API reads the pressure, temperature and humidity and gas data from the sensor, compensates the data and store it in the bme680_data structure instance passed by the user.

Detailed Description

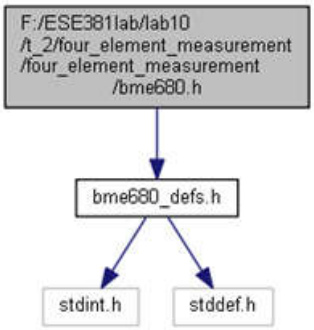
Sensor driver for BME680 sensor.

F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/bme680.h File Reference

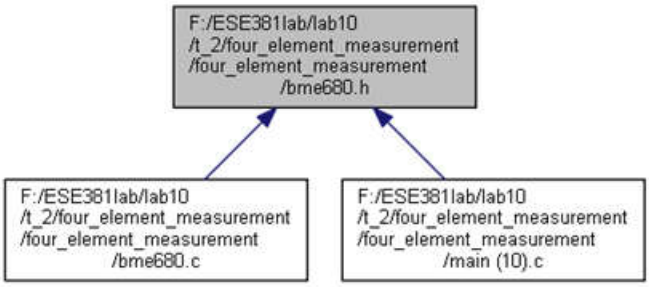
Sensor driver for BME680 sensor.

```
#include "bme680_defs.h"
```

Include dependency graph for bme680.h:



This graph shows which files directly or indirectly include this file:



Functions

- int8_t **bme680_init** (struct **bme680_dev** *dev)
  This API is the entry point. It reads the chip-id and calibration data from the sensor.

- int8_t **bme680_set_regs** (const uint8_t *reg_addr, const uint8_t *reg_data, uint8_t len, struct **bme680_dev** *dev)
  This API writes the given data to the register address of the sensor.

- int8_t **bme680_get_regs** (uint8_t reg_addr, uint8_t *reg_data, uint16_t len, struct **bme680_dev** *dev)
  This API reads the data from the given register address of the sensor.

- int8_t **bme680_soft_reset** (struct **bme680_dev** *dev)
  This API performs the soft reset of the sensor.

- int8_t **bme680_set_sensor_mode** (struct **bme680_dev** *dev)
  This API is used to set the power mode of the sensor.

- int8_t **bme680_get_sensor_mode** (struct **bme680_dev** *dev)
  This API is used to get the power mode of the sensor.

- void **bme680_set_profile_dur** (uint16_t duration, struct **bme680_dev** *dev)
  This API is used to set the profile duration of the sensor.

- void **bme680_get_profile_dur** (uint16_t *duration, const struct **bme680_dev** *dev)
  This API is used to get the profile duration of the sensor.

- int8_t **bme680_get_sensor_data** (struct **bme680_field_data** *data, struct **bme680_dev** *dev)
  This API reads the pressure, temperature and humidity and gas data from the sensor, compensates the data and store it in the bme680_data structure instance passed by the user.

- int8_t **bme680_set_sensor_settings** (uint16_t desired_settings, struct **bme680_dev** *dev)
  This API is used to set the oversampling, filter and T,P,H, gas selection settings in the sensor.

- int8_t **bme680_get_sensor_settings** (uint16_t desired_settings, struct **bme680_dev** *dev)
  This API is used to get the oversampling, filter and T,P,H, gas selection settings in the sensor.

Detailed Description

Sensor driver for BME680 sensor.

USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

The information provided is believed to be accurate and reliable. The copyright holder assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of the copyright holder.

Date

19 Jun 2018

Version

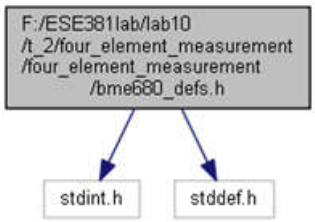
3.5.9

F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/bme680_defs.h File Reference

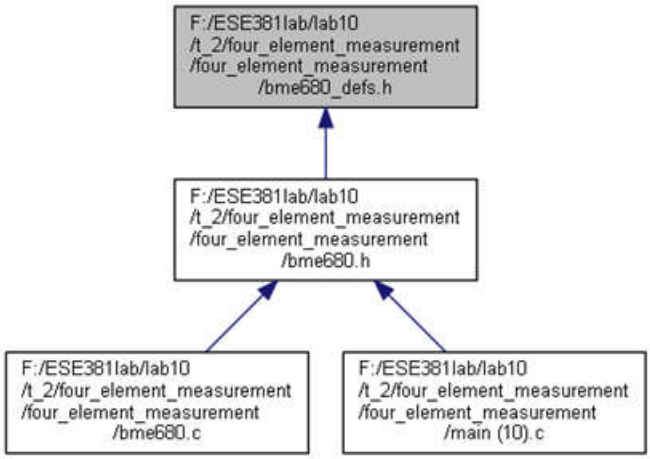
Sensor driver for BME680 sensor.

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

Include dependency graph for bme680_defs.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **bme680_field_data**
  Sensor field data structure.
- struct **bme680_calib_data**
  Structure to hold the Calibration data.
- struct **bme680_tph_sett**
  BME680 sensor settings structure which comprises of ODR, over-sampling and filter settings.
- struct **bme680_gas_sett**
  BME680 gas sensor which comprises of gas settings and status parameters.
- struct **bme680_dev**
  BME680 device structure.

Macros

Common macros

- #define **INT8_C**(x) S8_C(x)
- #define **UINT8_C**(x) U8_C(x)
- #define **INT16_C**(x) S16_C(x)
- #define **UINT16_C**(x) U16_C(x)
- #define **INT32_C**(x) S32_C(x)
- #define **UINT32_C**(x) U32_C(x)
- #define **INT64_C**(x) S64_C(x)
- #define **UINT64_C**(x) U64_C(x)

C standard macros

- #define **NULL** ((void *) 0)
- #define **BME680_POLL_PERIOD_MS** **UINT8_C**(10)
- #define **BME680_I2C_ADDR_PRIMARY** **UINT8_C**(0x76)
- #define **BME680_I2C_ADDR_SECONDARY** **UINT8_C**(0x77)
- #define **BME680_CHIP_ID** **UINT8_C**(0x61)
- #define **BME680_COEFF_SIZE** **UINT8_C**(41)
- #define **BME680_COEFF_ADDR1_LEN** **UINT8_C**(25)
- #define **BME680_COEFF_ADDR2_LEN** **UINT8_C**(16)
- #define **BME680_FIELD_LENGTH** **UINT8_C**(15)
- #define **BME680_FIELD_ADDR_OFFSET** **UINT8_C**(17)
- #define **BME680_SOFT_RESET_CMD** **UINT8_C**(0xb6)
- #define **BME680_OK** **INT8_C**(0)
- #define **BME680_E_NULL_PTR** **INT8_C**(-1)
- #define **BME680_E_COM_FAIL** **INT8_C**(-2)
- #define **BME680_E_DEV_NOT_FOUND** **INT8_C**(-3)
- #define **BME680_E_INVALID_LENGTH** **INT8_C**(-4)
- #define **BME680_W_DEFINE_PWR_MODE** **INT8_C**(1)
- #define **BME680_W_NO_NEW_DATA** **INT8_C**(2)
- #define **BME680_I_MIN_CORRECTION** **UINT8_C**(1)
- #define **BME680_I_MAX_CORRECTION** **UINT8_C**(2)
- #define **BME680_ADDR_RES_HEAT_VAL_ADDR** **UINT8_C**(0x00)
- #define **BME680_ADDR_RES_HEAT_RANGE_ADDR** **UINT8_C**(0x02)
- #define **BME680_ADDR_RANGE_SW_ERR_ADDR** **UINT8_C**(0x04)
- #define **BME680_ADDR_SENS_CONF_START** **UINT8_C**(0x5A)
- #define **BME680_ADDR_GAS_CONF_START** **UINT8_C**(0x64)
- #define **BME680_FIELD0_ADDR** **UINT8_C**(0x1d)
- #define **BME680_RES_HEAT0_ADDR** **UINT8_C**(0x5a)
- #define **BME680_GAS_WAIT0_ADDR** **UINT8_C**(0x64)
- #define **BME680_CONF_HEAT_CTRL_ADDR** **UINT8_C**(0x70)
- #define **BME680_CONF_ODR_RUN_GAS_NBC_ADDR** **UINT8_C**(0x71)
- #define **BME680_CONF_OS_H_ADDR** **UINT8_C**(0x72)
- #define **BME680_MEM_PAGE_ADDR** **UINT8_C**(0xf3)
- #define **BME680_CONF_T_P_MODE_ADDR** **UINT8_C**(0x74)
- #define **BME680_CONF_ODR_FILT_ADDR** **UINT8_C**(0x75)
- #define **BME680_COEFF_ADDR1** **UINT8_C**(0x89)
- #define **BME680_COEFF_ADDR2** **UINT8_C**(0xe1)
- #define **BME680_CHIP_ID_ADDR** **UINT8_C**(0xd0)
- #define **BME680_SOFT_RESET_ADDR** **UINT8_C**(0xe0)
- #define **BME680_ENABLE_HEATER** **UINT8_C**(0x00)
- #define **BME680_DISABLE_HEATER** **UINT8_C**(0x08)
- #define **BME680_DISABLE_GAS_MEAS** **UINT8_C**(0x00)
- #define **BME680_ENABLE_GAS_MEAS** **UINT8_C**(0x01)

- #define **BME680_OS_NONE UINT8_C**(0)
- #define **BME680_OS_1X UINT8_C**(1)
- #define **BME680_OS_2X UINT8_C**(2)
- #define **BME680_OS_4X UINT8_C**(3)
- #define **BME680_OS_8X UINT8_C**(4)
- #define **BME680_OS_16X UINT8_C**(5)
- #define **BME680_FILTER_SIZE_0 UINT8_C**(0)
- #define **BME680_FILTER_SIZE_1 UINT8_C**(1)
- #define **BME680_FILTER_SIZE_3 UINT8_C**(2)
- #define **BME680_FILTER_SIZE_7 UINT8_C**(3)
- #define **BME680_FILTER_SIZE_15 UINT8_C**(4)
- #define **BME680_FILTER_SIZE_31 UINT8_C**(5)
- #define **BME680_FILTER_SIZE_63 UINT8_C**(6)
- #define **BME680_FILTER_SIZE_127 UINT8_C**(7)
- #define **BME680_SLEEP_MODE UINT8_C**(0)
- #define **BME680_FORCED_MODE UINT8_C**(1)
- #define **BME680_RESET_PERIOD UINT32_C**(10)
- #define **BME680_MEM_PAGE0 UINT8_C**(0x10)
- #define **BME680_MEM_PAGE1 UINT8_C**(0x00)
- #define **BME680_HUM_REG_SHIFT_VAL UINT8_C**(4)
- #define **BME680_RUN_GAS_DISABLE UINT8_C**(0)
- #define **BME680_RUN_GAS_ENABLE UINT8_C**(1)
- #define **BME680_TMP_BUFFER_LENGTH UINT8_C**(40)
- #define **BME680_REG_BUFFER_LENGTH UINT8_C**(6)
- #define **BME680_FIELD_DATA_LENGTH UINT8_C**(3)
- #define **BME680_GAS_REG_BUF_LENGTH UINT8_C**(20)
- #define **BME680_OST_SEL UINT16_C**(1)
- #define **BME680_OSP_SEL UINT16_C**(2)
- #define **BME680_OSH_SEL UINT16_C**(4)
- #define **BME680_GAS_MEAS_SEL UINT16_C**(8)
- #define **BME680_FILTER_SEL UINT16_C**(16)
- #define **BME680_HCNTRL_SEL UINT16_C**(32)
- #define **BME680_RUN_GAS_SEL UINT16_C**(64)
- #define **BME680_NBCONV_SEL UINT16_C**(128)
- #define **BME680_GAS_SENSOR_SEL (BME680_GAS_MEAS_SEL | BME680_RUN_GAS_SEL | BME680_NBCONV_SEL)**
- #define **BME680_NBCONV_MIN UINT8_C**(0)
- #define **BME680_NBCONV_MAX UINT8_C**(10)
- #define **BME680_GAS_MEAS_MSK UINT8_C**(0x30)
- #define **BME680_NBCONV_MSK UINT8_C**(0X0F)
- #define **BME680_FILTER_MSK UINT8_C**(0X1C)
- #define **BME680_OST_MSK UINT8_C**(0XE0)
- #define **BME680_OSP_MSK UINT8_C**(0X1C)
- #define **BME680_OSH_MSK UINT8_C**(0X07)
- #define **BME680_HCTRL_MSK UINT8_C**(0x08)
- #define **BME680_RUN_GAS_MSK UINT8_C**(0x10)
- #define **BME680_MODE_MSK UINT8_C**(0x03)
- #define **BME680_RHRANGE_MSK UINT8_C**(0x30)
- #define **BME680_RSERROR_MSK UINT8_C**(0xf0)
- #define **BME680_NEW_DATA_MSK UINT8_C**(0x80)
- #define **BME680_GAS_INDEX_MSK UINT8_C**(0x0f)
- #define **BME680_GAS_RANGE_MSK UINT8_C**(0x0f)
- #define **BME680_GASM_VALID_MSK UINT8_C**(0x20)
- #define **BME680_HEAT_STAB_MSK UINT8_C**(0x10)
- #define **BME680_MEM_PAGE_MSK UINT8_C**(0x10)
- #define **BME680_SPI_RD_MSK UINT8_C**(0x80)
- #define **BME680_SPI_WR_MSK UINT8_C**(0x7f)

- #define **BME680_BIT_H1_DATA_MSK UINT8_C**(0x0F)
- #define **BME680_GAS_MEAS_POS UINT8_C**(4)
- #define **BME680_FILTER_POS UINT8_C**(2)
- #define **BME680_OST_POS UINT8_C**(5)
- #define **BME680_OSP_POS UINT8_C**(2)
- #define **BME680_RUN_GAS_POS UINT8_C**(4)
- #define **BME680_T2_LSB_REG** (1)
- #define **BME680_T2_MSB_REG** (2)
- #define **BME680_T3_REG** (3)
- #define **BME680_P1_LSB_REG** (5)
- #define **BME680_P1_MSB_REG** (6)
- #define **BME680_P2_LSB_REG** (7)
- #define **BME680_P2_MSB_REG** (8)
- #define **BME680_P3_REG** (9)
- #define **BME680_P4_LSB_REG** (11)
- #define **BME680_P4_MSB_REG** (12)
- #define **BME680_P5_LSB_REG** (13)
- #define **BME680_P5_MSB_REG** (14)
- #define **BME680_P7_REG** (15)
- #define **BME680_P6_REG** (16)
- #define **BME680_P8_LSB_REG** (19)
- #define **BME680_P8_MSB_REG** (20)
- #define **BME680_P9_LSB_REG** (21)
- #define **BME680_P9_MSB_REG** (22)
- #define **BME680_P10_REG** (23)
- #define **BME680_H2_MSB_REG** (25)
- #define **BME680_H2_LSB_REG** (26)
- #define **BME680_H1_LSB_REG** (26)
- #define **BME680_H1_MSB_REG** (27)
- #define **BME680_H3_REG** (28)
- #define **BME680_H4_REG** (29)
- #define **BME680_H5_REG** (30)
- #define **BME680_H6_REG** (31)
- #define **BME680_H7_REG** (32)
- #define **BME680_T1_LSB_REG** (33)
- #define **BME680_T1_MSB_REG** (34)
- #define **BME680_GH2_LSB_REG** (35)
- #define **BME680_GH2_MSB_REG** (36)
- #define **BME680_GH1_REG** (37)
- #define **BME680_GH3_REG** (38)
- #define **BME680_REG_FILTER_INDEX UINT8_C**(5)
- #define **BME680_REG_TEMP_INDEX UINT8_C**(4)
- #define **BME680_REG_PRES_INDEX UINT8_C**(4)
- #define **BME680_REG_HUM_INDEX UINT8_C**(2)
- #define **BME680_REG_NBCONV_INDEX UINT8_C**(1)
- #define **BME680_REG_RUN_GAS_INDEX UINT8_C**(1)
- #define **BME680_REG_HCTRL_INDEX UINT8_C**(0)
- #define **BME680_MAX_OVERFLOW_VAL INT32_C**(0x40000000)
- #define **BME680_CONCAT_BYTES**(msb, lsb) (((uint16_t)msb << 8) | (uint16_t)lsb)
- #define **BME680_SET_BITS**(reg_data, bitname, data)
- #define **BME680_GET_BITS**(reg_data, bitname)
- #define **BME680_SET_BITS_POS_0**(reg_data, bitname, data)
- #define **BME680_GET_BITS_POS_0**(reg_data, bitname) (reg_data & (bitname##_MSK))
- enum **bme680_intf** { **BME680_SPI_INTF**, **BME680_I2C_INTF** }
  Interface selection Enumerations.

- typedef int8_t(* **bme680_com_fptr_t**) (uint8_t dev_id, uint8_t reg_addr, uint8_t *data, uint16_t len)
- typedef void(* **bme680_delay_fptr_t**) (uint32_t period)

Detailed Description

Sensor driver for BME680 sensor.

Date

19 Jun 2018

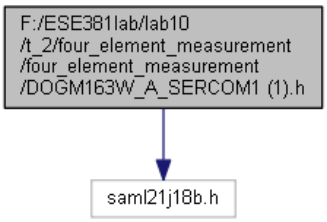
Version

3.5.9

F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/DOGM163W_A_SERCOM1 (1).h File Reference

```
#include "saml21j18b.h"
```

Include dependency graph for DOGM163W_A_SERCOM1 (1).h:



Functions

- void **init_lcd_dog** (void)
- void **update_lcd_dog** (void)

Variables

- char **dsp_buff1** [17]
- char **dsp_buff2** [17]
- char **dsp_buff3** [17]

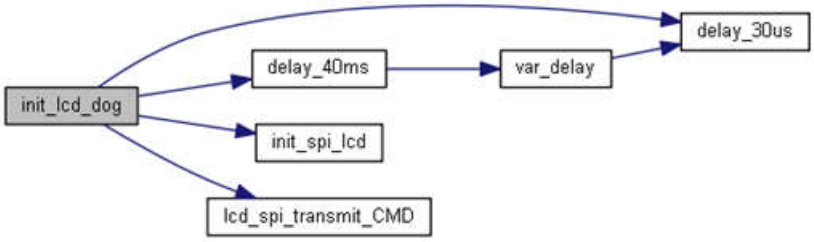
Function Documentation

void init_lcd_dog (void)

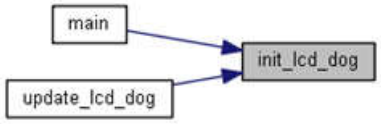
Initializes LCD screen

Definition at line 98 of file DOGM163W_A_SERCOM1 (2).c.

Here is the call graph for this function:



Here is the caller graph for this function:

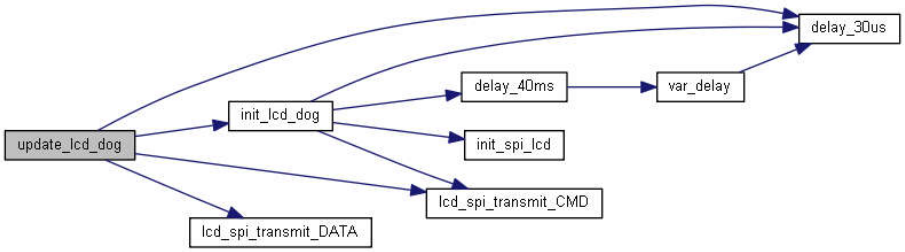


void update_lcd_dog (void)

Updates lcd screen using display buffers

Definition at line 125 of file DOGM163W_A_SERCOM1 (2).c.

Here is the call graph for this function:



Variable Documentation

char dsp_buff1[17]

DOGM163W_A_SERCOM1.h

Created: 4/7/2020 1:27:27 PM Author : Emmanuel Benard, Kaleb Croft

Definition at line 14 of file DOGM163W_A_SERCOM1 (2).c.

char dsp_buff2[17]

Definition at line 15 of file DOGM163W_A_SERCOM1 (2).c.

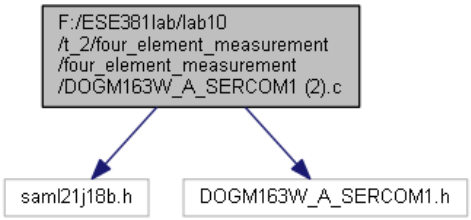
char dsp_buff3[17]

Definition at line 16 of file DOGM163W_A_SERCOM1 (2).c.

F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/DOGM163W_A_SERCOM1 (2).c File Reference

```
#include "saml21j18b.h"
#include "DOGM163W_A_SERCOM1.h"
```

Include dependency graph for DOGM163W_A_SERCOM1 (2).c:



Macros

- #define **FREQUENCY** 4

Functions

- void **init_spi_lcd** (void)
- void **lcd_spi_transmit_CMD** (unsigned char cmd)
- void **lcd_spi_transmit_DATA** (unsigned char data)
- void **delay_30us** (void)
- void **var_delay** (int delay_var)
- void **delay_40ms** (void)
- void **init_lcd_dog** (void)
- void **update_lcd_dog** (void)

Variables

- char **dsp_buff1** [17]
- char **dsp_buff2** [17]
- char **dsp_buff3** [17]

Macro Definition Documentation

#define FREQUENCY 4

DOGM163W_A_SERCOM1.c

Created: 4/7/2020 1:52:32 PM Author: Emmanuel Benard, Kaleb Croft

Definition at line 11 of file DOGM163W_A_SERCOM1 (2).c.

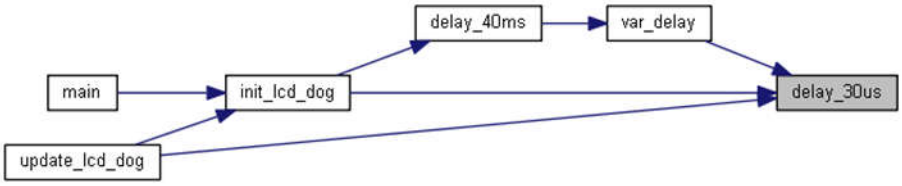
Function Documentation

void delay_30us (void)

Creates a delay of 30 microseconds

Definition at line 69 of file DOGM163W_A_SERCOM1 (2).c.

Here is the caller graph for this function:



void delay_40ms (void)

Creates a delay of 40 milliseconds

Definition at line 91 of file DOGM163W_A_SERCOM1 (2).c.

Here is the call graph for this function:

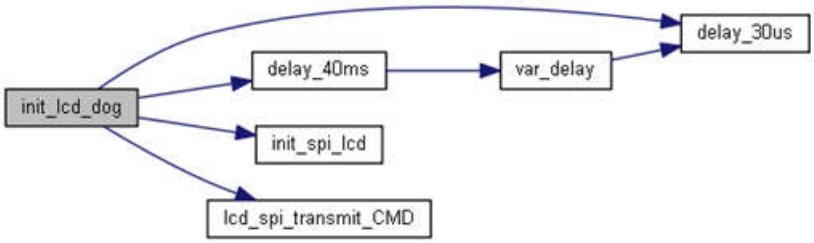
Here is the caller graph for this function:

void init_lcd_dog (void)

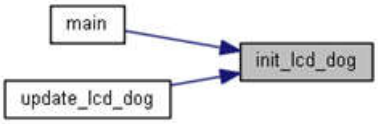
Initializes LCD screen

Definition at line 98 of file DOGM163W_A_SERCOM1 (2).c.

Here is the call graph for this function:



Here is the caller graph for this function:

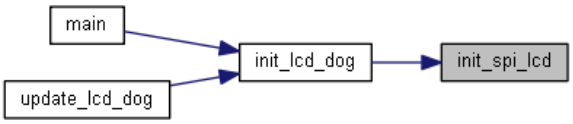


void init_spi_lcd (void)

Initializes SERCOM port for LCD screen

Definition at line 21 of file DOGM163W_A_SERCOM1 (2).c.

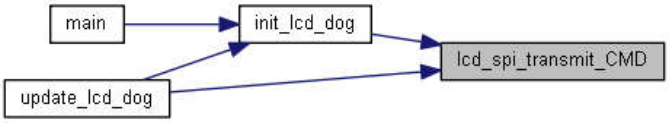
Here is the caller graph for this function:



void lcd_spi_transmit_CMD (unsigned char *cmd*)

Definition at line 49 of file DOGM163W_A_SERCOM1 (2).c.

Here is the caller graph for this function:

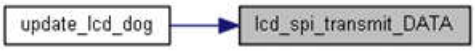


void lcd_spi_transmit_DATA (unsigned char *data*)

Transmits a data byte to the LCD screen

Definition at line 59 of file DOGM163W_A_SERCOM1 (2).c.

Here is the caller graph for this function:

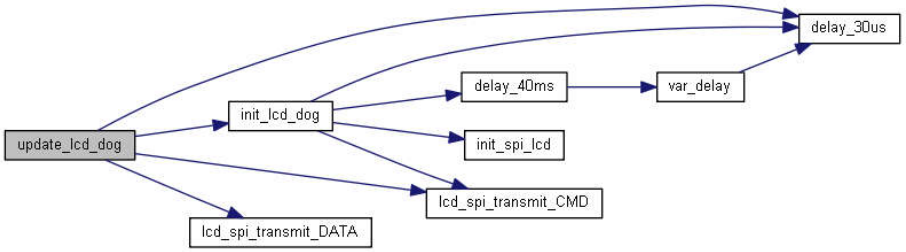


void update_lcd_dog (void)

Updates lcd screen using display buffers

Definition at line 125 of file DOGM163W_A_SERCOM1 (2).c.

Here is the call graph for this function:



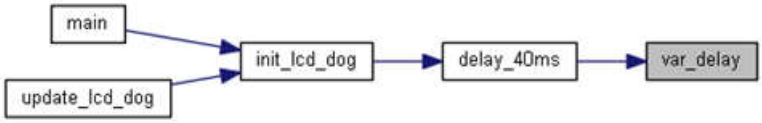
void var_delay (int *delay_var*)

This procedure will generate a variable delay for a fixed period of time based on the passed value

Definition at line 80 of file DOGM163W_A_SERCOM1 (2).c.

Here is the call graph for this function:

Here is the caller graph for this function:



Variable Documentation

char dsp_buff1[17]

DOGM163W_A_SERCOM1.h

Created: 4/7/2020 1:27:27 PM Author : Emmanuel Benard, Kaleb Croft

Definition at line 14 of file DOGM163W_A_SERCOM1 (2).c.

char dsp_buff2[17]

Definition at line 15 of file DOGM163W_A_SERCOM1 (2).c.

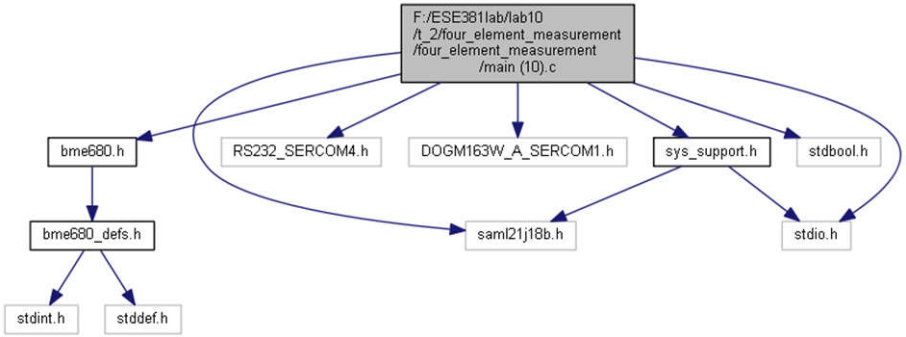
char dsp_buff3[17]

Definition at line 16 of file DOGM163W_A_SERCOM1 (2).c.

F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/main (10).c File Reference

```
#include "saml21j18b.h"
#include "bme680.h"
#include "RS232_SERCOM4.h"
#include "DOGM163W_A_SERCOM1.h"
#include <stdio.h>
#include <stdbool.h>
#include "sys_support.h"
```

Include dependency graph for main (10).c:



Macros

- #define **DUMMY_VAL** 0x00

Functions

- int8_t **user_spi_read** (uint8_t dev_id, uint8_t reg_addr, uint8_t *reg_data, uint16_t len)
- int8_t **user_spi_write** (uint8_t dev_id, uint8_t reg_addr, uint8_t *reg_data, uint16_t len)
- void **user_delay_ms** (uint32_t period)
- int **main** (void)

Variables

- unsigned char * **ARRAY_PORT_PINCFG0**
- unsigned char * **ARRAY_PORT_PMUX0**
- unsigned char * **ARRAY_PORT_PINCFG1**
- unsigned char * **ARRAY_PORT_PMUX1**

Macro Definition Documentation

#define DUMMY_VAL 0x00

main.c

Created: 5/6/2020 12:01:25 PM Author : Emmanuel Benard, Kaleb Croft

Definition at line 16 of file main (10).c.

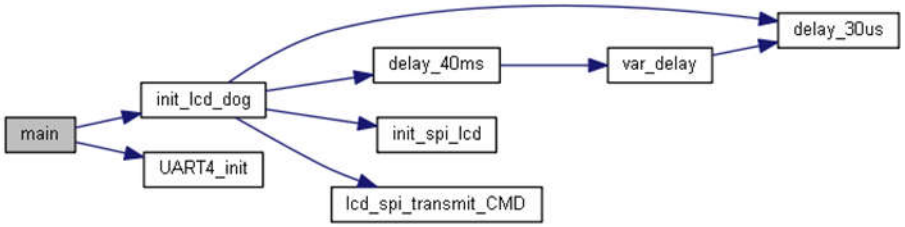
Function Documentation

int main (void)

Main function for task 2. Reads temperature, pressure, humidity, and gas from the BME680. Outputs the values to the LCD screen.

Definition at line 183 of file main (10).c.

Here is the call graph for this function:



void user_delay_ms (uint32_t *period*)

Wait for a period amount of milliseconds

Definition at line 169 of file main (10).c.

int8_t user_spi_read (uint8_t *dev_id*, uint8_t *reg_addr*, uint8_t * *reg_data*, uint16_t *len*)

Passes address and dereferenced dummy data pointer(s) (# of pointers depending on len) to spi transfer () Param dev_id and Return val unused

Definition at line 54 of file main (10).c.

int8_t user_spi_write (uint8_t *dev_id*, uint8_t *reg_addr*, uint8_t * *reg_data*, uint16_t *len*)

Passes address and dereferenced data pointer(s) (# of pointers depending on len) to spi transfer () Param dev_id and Return val unused

Definition at line 74 of file main (10).c.

Variable Documentation

unsigned char* ARRAY_PORT_PINCFG0

Definition at line 18 of file main (10).c.

unsigned char* ARRAY_PORT_PINCFG1

Definition at line 20 of file main (10).c.

unsigned char* ARRAY_PORT_PMUX0

Definition at line 19 of file main (10).c.

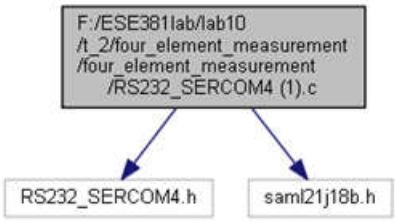
unsigned char* ARRAY_PORT_PMUX1

Definition at line 21 of file main (10).c.

F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/RS232_SERCOM4 (1).c File Reference

```
#include "RS232_SERCOM4.h"
#include "saml21j18b.h"
```

Include dependency graph for RS232_SERCOM4 (1).c:



Functions

- void **UART4_init** (void)
- void **UART4_write** (char data)
- char **UART4_read** (void)

Variables

- unsigned char * **ARRAY_PORT_PINCFG1** = (unsigned char*)&REG_PORT_PINCFG1
- unsigned char * **ARRAY_PORT_PMUX1** = (unsigned char*)&REG_PORT_PMUX1

Function Documentation

void UART4_init (void)

initialize UART4 to transmit at 9600 Baud

Definition at line 24 of file RS232_SERCOM4 (1).c.

Here is the caller graph for this function:

char UART4_read (void)

Read a data byte from UART4

Definition at line 52 of file RS232_SERCOM4 (1).c.

void UART4_write (char *data*)

Send a data byte to UART4

Definition at line 44 of file RS232_SERCOM4 (1).c.

Variable Documentation

unsigned char* ARRAY_PORT_PINCFG1 = (unsigned char*)&REG_PORT_PINCFG1

RS232_SERCOM4.c

Created: 4/7/2020 12:45:56 PM Author : Emmanuel Benard, Kaleb Croft

Definition at line 12 of file RS232_SERCOM4 (1).c.

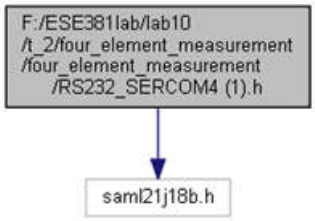
unsigned char* ARRAY_PORT_PMUX1 = (unsigned char*)&REG_PORT_PMUX1

Definition at line 13 of file RS232_SERCOM4 (1).c.

F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/RS232_SERCOM4 (1).h File Reference

```
#include "saml21j18b.h"
```

Include dependency graph for RS232_SERCOM4 (1).h:



Functions

- void **UART4_init** (void)
- void **UART4_write** (char data)
- char **UART4_read** (void)

Variables

- unsigned char * **ARRAY_PORT_PINCFG1**
- unsigned char * **ARRAY_PORT_PMUX1**

Function Documentation

void UART4_init (void)

initialize UART4 to transmit at 9600 Baud

Definition at line 24 of file RS232_SERCOM4 (1).c.

Here is the caller graph for this function:

char UART4_read (void)

Read a data byte from UART4

Definition at line 52 of file RS232_SERCOM4 (1).c.

void UART4_write (char *data*)

Send a data byte to UART4

Definition at line 44 of file RS232_SERCOM4 (1).c.

Variable Documentation

unsigned char* ARRAY_PORT_PINCFG1

RS232_SERCOM4.h

Created: 4/7/2020 1:26:38 PM Author: Emmanuel Benard, Kaleb Croft

Definition at line 13 of file RS232_SERCOM4 (1).h.

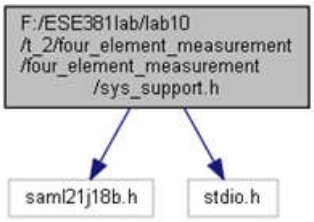
unsigned char* ARRAY_PORT_PMUX1

Definition at line 14 of file RS232_SERCOM4 (1).h.

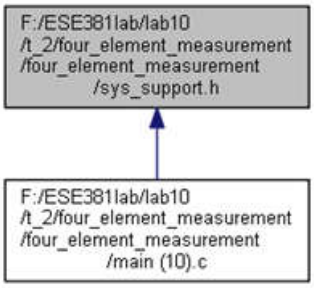
F:/ESE381lab/lab10/t_2/four_element_measurement/four_element_measurement/sys_support.h File Reference

```
#include "saml21j18b.h"
#include <stdio.h>
```

Include dependency graph for sys_support.h:



This graph shows which files directly or indirectly include this file:



Functions

- int **_write** (FILE *f, char *buf, int n)
- int **_read** (FILE *f, char *buf, int n)
- int **_close** (FILE *f)
- int **_fstat** (FILE *f, void *p)
- int **_isatty** (FILE *f)
- int **_lseek** (FILE *f, int o, int w)
- void * **_sbrk** (int i)

Function Documentation

int _close (FILE * *f*)

int _fstat (FILE * *f*, void * *p*)

int _isatty (FILE * *f*)

int _lseek (FILE * *f*, int *o*, int *w*)

int _read (FILE * *f*, char * *buf*, int *n*)

void* _sbrk (int *i*)

int _write (FILE * *f*, char * *buf*, int *n*)

Index

INDEX