
BMP180 STM32 Driver

Ceyhun Şen

Jul 08, 2022

CONTENTS

1	License	3
2	Index	5
2.1	Getting Started	5
2.1.1	Adding to Your Project	5
2.1.2	Units	5
2.1.3	Simple Usage	5
2.1.4	More On Getting Sensor Data	6
2.1.5	Oversampling Settings	6
2.1.6	Sea Pressure	7
2.2	API Reference	7
	Index	11

Welcome to the documentation of STM32 driver for BMP180 barometric pressure/temperature/altitude sensor.
Source code is available at [Github](#).

LICENSE

MIT License

Copyright (c) 2022 Ceyhun Şen

Permission **is** hereby granted, free of charge, to **any** person obtaining a copy of this software **and** associated documentation files (the "**Software**"), to deal **in** the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, **and/or** sell copies of the Software, **and** to permit persons to whom the Software **is** furnished to do so, subject to the following conditions:

The above copyright notice **and** this permission notice shall be included **in all** copies **or** substantial portions of the Software.

THE SOFTWARE IS PROVIDED "**AS IS**", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2.1 Getting Started

2.1.1 Adding to Your Project

1. Copy `bmp180` directory to your projects `drivers` directory.
2. Add `bmp180.c` file to your projects source files.
3. Add `bmp180/inc/` directory to your projects include path.
4. Change `I2C_LIB` definition in `bmp180/inc/bmp180.h` (line 14) to your MCU's I2C HAL library (e.g. `"stm32f4xx_hal.h"`).

2.1.2 Units

- Temperature: Celsius
- Pressure: Pascal
- Altitude: Meter

2.1.3 Simple Usage

```
// Define bmp180_t struct instance
bmp180_t bmp180 = {.oversampling_setting = standart};

// Wait till initialization is complete
while (bmp180_init(&hi2c1, &bmp180));

// Get all the values
bmp180_get_all(&bmp180);
float temperature = bmp180.temperature;
int32_t pressure = bmp180.pressure;
float altitude = bmp180.altitude;
```

2.1.4 More On Getting Sensor Data

There is actually 2 sensor data: temperature and pressure. Altitude is calculated from pressure data. And each one of them can be measured separately. But each one of them needs some other data from other calculations.

- Pressure data needs some calibration values from temperature measurements.
- Altitude calculation needs pressure data.

There are separate functions for all 3 of them but measuring them separately may result in inaccurate data. So using `bmp180_get_all()` function is the recommended way. But if you really need to measure them separately, here is the recommended minimums of function calls:

Temperature

```
bmp180_get_temperature(&bmp180);  
float temperature = bmp180.temperature;
```

Pressure

```
bmp180_get_temperature(&bmp180);  
bmp180_get_pressure(&bmp180);  
int32_t pressure = bmp180.pressure;
```

Altitude

```
bmp180_get_temperature(&bmp180);  
bmp180_get_pressure(&bmp180);  
bmp180_get_altitude(&bmp180);  
float altitude = bmp180.altitude;
```

2.1.5 Oversampling Settings

BMP180 offers hardware oversampling for sensor data. These are ultra low power, standard, high resolution and ultra high resolution. Check BMP180's datasheet for detailed information about oversampling.

Changing Oversampling Setting

Oversampling setting is stored in the `bmp180_t` struct. If you want to change oversampling setting, you should change `oversampling_setting` member and call `bmp180_init()` function.

For example if you want to change it to ultra high resolution:

```
bmp180.oversampling_setting = ultra_high_resolution;  
bmp180_init(&hi2c1, &bmp180);
```

Warning: If you don't call `bmp180_init()` function after changing setting, oversampling won't change.

2.1.6 Sea Pressure

Default sea pressure is 101325 pascal.

Changing Sea Pressure

Sea pressure can be changed with modifying `sea_pressure` member of `bmp180_t` struct or calling `bmp180_set_sea_spressure()`.

```
bmp180_set_sea_pressure(&bmp180, 101400);
```

If you want to measure altitude from any take-off point, first measure pressure at the ground and set it as sea pressure. After that, the new altitude calculation is your altitude from ground.

```
// ...
// Getting pressure and setting it as sea pressure
bmp180_get_all(&bmp180);
bmp180_set_sea_pressure(&bmp180, bmp180.pressure);
// After take-off, measure altitude
bmp180_get_all(&bmp180);
float higher_altitude_than_ground = bmp180.altitude;
// ...
```

I2C Interface

BMP180 sensor only supports I2C interface. So, this driver uses STM32's I2C HAL libraries. If you want to change it to LL drivers, modify `bmp180_read()`, `bmp180_write()` and `bmp180_is_ready()` functions.

2.2 API Reference

Author

Ceyhun Şen

Defines

I2C_LIB

Library that includes I2C functions.

Change this definition to your MCU's I2C HAL library. E.g. "stm32f4xx_hal.h".

Typedefs

typedef struct *bmp180_t* **bmp180_t**

Enums

enum **_bmp180_oversampling_settings**

Oversampling settings for BMP180 sensor.

Values:

enumerator **ultra_low_power**

enumerator **standart**

enumerator **high_resolution**

enumerator **ultra_high_resolution**

Functions

uint8_t **bmp180_init**(I2C_HandleTypeDef *hi2cx, *bmp180_t* *bmp180)

Initialize sensor and get calibration values.

Parameters

- **hi2cx** – I2C handle.
- **bmp180** – *bmp180_t* struct to initialize.

Returns

0 on success, 1 on sensor is not ready, 2 on sensor error.

void **bmp180_get_all**(*bmp180_t* *bmp180)

Get all sensor data at once.

Parameters

bmp180 – *bmp180_t* struct to write data.

Return values

None. –

void **bmp180_get_temperature**(*bmp180_t* *bmp180)

Get temperature data.

Parameters

bmp180 – *bmp180_t* struct to write data.

Return values

None. –

void **bmp180_get_pressure**(*bmp180_t* *bmp180)

Get pressure data.

Parameters

bmp180 – *bmp180_t* struct to write data.

Return values

None. –

void **bmp180_get_altitude**(*bmp180_t* *bmp180)

Calculate altitude from pressure data.

Parameters

bmp180 – *bmp180_t* struct to write data.

Return values

None. –

void **bmp180_set_sea_pressure**(*bmp180_t* *bmp180, int32_t sea_pressure)

Set sea pressure.

Parameters

- **bmp180** – *bmp180_t* struct to write data.
- **sea_pressure** – New sea pressure.

Return values

None. –

struct **bmp180_t**

#include <bmp180.h> Holds sensor data, sensor settings and calibration values.

Public Members

I2C_HandleTypeDef ***hi2cx**

float **temperature**

int32_t **pressure**

float **altitude**

int32_t **sea_pressure**

enum *_bmp180_oversampling_settings* **oversampling_setting**

uint8_t **oss**

int16_t **AC1**

`int16_t AC2`

`int16_t AC3`

`uint16_t AC4`

`uint16_t AC5`

`uint16_t AC6`

`int16_t B1`

`int16_t B2`

`int32_t B3`

`uint32_t B4`

`int32_t B5`

`int32_t B6`

`uint32_t B7`

`int16_t MB`

`int16_t MC`

`int16_t MD`

INDEX

Symbols

`_bmp180_oversampling_settings` (C++ *enum*), 8
`_bmp180_oversampling_settings::high_resolution`
(C++ *enumerator*), 8
`_bmp180_oversampling_settings::standart`
(C++ *enumerator*), 8
`_bmp180_oversampling_settings::ultra_high_resolution`
(C++ *enumerator*), 8
`_bmp180_oversampling_settings::ultra_low_power`
(C++ *enumerator*), 8
`bmp180_t::temperature` (C++ *member*), 9
`I2C_LIB` (C *macro*), 7

B

`bmp180_get_all` (C++ *function*), 8
`bmp180_get_altitude` (C++ *function*), 9
`bmp180_get_pressure` (C++ *function*), 8
`bmp180_get_temperature` (C++ *function*), 8
`bmp180_init` (C++ *function*), 8
`bmp180_set_sea_pressure` (C++ *function*), 9
`bmp180_t` (C++ *struct*), 9
`bmp180_t` (C++ *type*), 8
`bmp180_t::AC1` (C++ *member*), 9
`bmp180_t::AC2` (C++ *member*), 9
`bmp180_t::AC3` (C++ *member*), 10
`bmp180_t::AC4` (C++ *member*), 10
`bmp180_t::AC5` (C++ *member*), 10
`bmp180_t::AC6` (C++ *member*), 10
`bmp180_t::altitude` (C++ *member*), 9
`bmp180_t::B1` (C++ *member*), 10
`bmp180_t::B2` (C++ *member*), 10
`bmp180_t::B3` (C++ *member*), 10
`bmp180_t::B4` (C++ *member*), 10
`bmp180_t::B5` (C++ *member*), 10
`bmp180_t::B6` (C++ *member*), 10
`bmp180_t::B7` (C++ *member*), 10
`bmp180_t::hi2cx` (C++ *member*), 9
`bmp180_t::MB` (C++ *member*), 10
`bmp180_t::MC` (C++ *member*), 10
`bmp180_t::MD` (C++ *member*), 10
`bmp180_t::oss` (C++ *member*), 9
`bmp180_t::oversampling_setting` (C++ *member*), 9
`bmp180_t::pressure` (C++ *member*), 9
`bmp180_t::sea_pressure` (C++ *member*), 9