Pawel Gburzynski

# BMP280 pressure/temp

## driver

September 20, 2017

## Introduction

BMP280 is an air pressure/temperature combination sensor manufactured by Bosch Sensortec. Documentation available from the manufacturer (data sheet + specification) covers all the technical intricacies. That documentation is not required to understand how to use the driver and what you can expect from the sensor.

## Summary of the sensor's functionality

The sensor is reasonably simple in terms of its functionality perceived by the application. It generates no events and only provides readouts of air pressure and temperature. The temperature is required to correctly transform the raw pressure sample into the actual pressure. The transformation formula is rather complicated and involves (in addition to the temperature reading) twelve 16-bit values stored in the sensor's EPROM. The transformation is not automatic, i.e., it has to be carried out by the driver. The calculations affect both the temperature and the pressure.

The sensor operates in two active modes (in addition to the inactive sleep mode): *normal* and *forced*. In the normal mode, the sensor continuously carries out the measurements at the rate determined by three parameters (as described below). In the forced mode, the sensor remains dormant (effectively in the sleep state) until explicitly requested to take a measurement. Then it does what it has been asked for and goes back to sleep. The forced mode (with reasonably sparse readings) results in the lowest average current drain.

The three parameters/features trading the frequency and accuracy of measurements for the current drain are:

1. oversampling, i.e., how many takes are averaged into a sample
2. filtering, i.e., smoothing out the results based on the history (using exponential moving average)
3. standby, i.e., idle intervals inserted between measurements

Oversampling means making a single sample the average of a number of rapidly collected samples/takes. The oversampling parameter, settable independently for temperature and pressure, affects the amount of time needed to collect a sample as well as the accuracy (or rather resolution) of the sample, as summarized in this table:

| Oversampling | Pressure resolution | Temperature resolution |
|---|---|---|
| x1 | 16 bit / 2.62 Pa | 16 bit / 0.0050 °C |
| x2 | 17 bit / 1.31 Pa | 17 bit / 0.0025 °C |
| x4 | 18 bit / 0.66 Pa | 18 bit / 0.0012 °C |
| x8 | 19 bit / 0.33 Pa | 19 bit / 0.0006 °C |
| x16 | 20 bit / 0.16 Pa | 20 bit / 0.0003 °C |

The data sheet recommends using the temperature oversampling of x2 for the x16 setting for pressure. A higher oversampling setting for temperature is possible, but useless, unless (for some reason) a high resolution of the temperature itself is needed. For the remaining pressure settings, the x1 setting for the temperature (no oversampling) should do. Needless to say resolution is not the same as accuracy.

With the filtering on, a single returned sample is calculated as a moving average of the new (raw) sample and the previous (presented) sample. This is applied after oversampling. The formula is:

*data_filtered = (data_filtered_old x (f – 1) + data_acquired) / f*

where *f* is the coefficient whose value can be 1 (filtering off), 2, 4, 8, or 16.

Standby is the amount of idle time inserted between two sample collections (after oversampling). This only applies to the normal mode, because in the forced mode only one sample is issued at a time, and the frequency of sampling is determined by the application. The available discrete settings in milliseconds are: 0.5, 62.5, 125, 250, 500, 1000, 2000, and 4000. For a longer standby interval, the effective sampling rate is practically determined by the interval. The actual sample collection time is about 5.5 ms (no oversampling, both pressure and temperature combined). This value has to be (roughly) multiplied by the oversampling factor, to yield the total time for taking the sample, and then added to the standby time to produce the inter-sample interval and thus the frequency of sampling.

The sensor drains 0.3 µA when sleeping. This is also the current drain in forced mode between explicit sample acquisitions. While the sensor is waiting for standby in the normal mode, this value raises to the still insignificant 0.5 µA. Some idea about the effective current drain is given in the following table for 1Hz data acquisition in forced mode:

| Oversampling | Average current (µA) |
|---|---|
| Px1, Tx1 | 2.74 – 4.16 |
| Px2, Tx1 | 4.17 – 6.27 |
| Px4, Tx1 | 7.02 – 10.50 |
| Px8, Tx1 | 12.70 – 18.95 |
| Px16, Tx2 | 24.8 – 36.85 |

## The interface

The sensor must be turned on to become active. Here is the operation:

```
bmp280_on (word mode);
```

where **mode** is a collection of flags defined in *bmp280.h*. which can be combined (or'red) to set the sensor up as needed by the application.

The primary mode is set with these constants:

```
BMP280_MODE_NORMAL
BMP280_MODE_FORCED
```

where the second one can be skipped (the forced mode is then assumed by default).

Oversampling is set up with these constants (one for pressure and one for temperature):

```
BMP280_MODE_PRESS_OVS_OFF
BMP280_MODE_PRESS_OVS_1X
BMP280_MODE_PRESS_OVS_2X
BMP280_MODE_PRESS_OVS_4X
BMP280_MODE_PRESS_OVS_8X
BMP280_MODE_PRESS_OVS_16X

BMP280_MODE_TEMP_OVS_OFF
BMP280_MODE_TEMP_OVS_1X
BMP280_MODE_TEMP_OVS_2X
BMP280_MODE_TEMP_OVS_4X
BMP280_MODE_TEMP_OVS_8X
BMP280_MODE_TEMP_OVS_16X
```

The **OFF** constants turn off the corresponding measure (so it won't be returned as part of the sensor data). While formally possible, it makes no sense to select **OFF** for both temperature and pressure, because then **read_sensor** (see below) will return no data.

Here are the constants for setting up the filter (which is off by default):

```
BMP280_MODE_FILTER_OFF
BMP280_MODE_FILTER_2
BMP280_MODE_FILTER_4
BMP280_MODE_FILTER_8
BMP280_MODE_FILTER_16
```

The last set of constants is used to define the standby time (which by default is set to the lowest value, 0.5 ms):

```
BMP280_MODE_STANDBY_05_MS
BMP280_MODE_STANDBY_63_MS
BMP280_MODE_STANDBY_125_MS
BMP280_MODE_STANDBY_250_MS
BMP280_MODE_STANDBY_500_MS
BMP280_MODE_STANDBY_1000_MS
BMP280_MODE_STANDBY_2000_MS
BMP280_MODE_STANDBY_4000_MS
```

If the sensor is powered from a pin, the operation will set the pin high and actually deliver voltage to the sensor. If the sensor has been wired this way, its current drain in the off state will be precisely zero.

For example:

```
bmp280_on (BMP280_MODE_PRESS_OVS_2X |
           BMP280_MODE_FILTER_4 |
           BMP280_MODE_STANDBY_500_MS);
```

sets the sensor up in the forced (default mode), to return the pressure only, oversampled twice at 0.5 s intervals, and filtered with the factor of 4.

The operation to turn the sensor off is:

```
bmp280_off ();
```



Note that if the sensor is not wired to be powered dynamically (from a pin) and has been set up in the forced mode, turning it off is purely logical and doesn't reduce the current drain.

The data is acquired in the standard way:

```
read_sensor (word st, address val);
```

Both values, i.e., pressure and temperature, are 32-bit integer numbers. If both readings have been selected (neither is **OFF**), **val** must point to an array of 4 **words** interpreted as two consecutive **lwords**. The first **lword** will be filled with the pressure in Pa (this number is never negative), and the second one will be filled with the temperature in degrees Celsius times 100 (the returned value must be divided by 100 to yield the number of degrees). The range is from -40 to +85 for the temperature and 0 to 110000 Pa for the pressure.

If any of the readings is **OFF**, only one **lword** pointed to by val will be filled. If both readings happen to be **OFF**, nothing will be stored under **val**.

In the forced mode, the operation has to wait for the completion of the explicitly started measurement, so the state argument is useful. If **st** is **WNONE**, the function will busy-wait for the end conversion.

In the normal mode, the state argument is not used, because the function always finds the data ready. The sensor is equipped with shadow registers which make it possible to read (old) data while new data is being collected.

Note that the operation carries out the compensation prescribed by Sensortec which is non-trivial and takes a few divisions and multiplications.