AN12519

Implementing post-assembly TPMS sensor calibration

Rev. 1 — 1 July 2019 Application note

1 Introduction

Sensor calibration can be performed when the sensor tolerance is too large compared to the project's requirements. Calibration minimizes the sensor deviation in a certain portion of the sensor's operating range.

Sensor deviation includes sensitivity and offset errors, with the offset error as the most significant contributor to the total error. After the TPMS board has been placed in the housing, implementing sensor calibration on the finished module enables error correction for chip level errors (specified in the data sheet) and errors resulting from the stress of the assembly process (e.g., soldering, housing).

Performing sensitivity calibration in addition to offset calibration should not be required to meet the tolerance requirements of the project. The sensitivity errors usually remain negligible compared to the offset errors or other sources of error. For this reason, this document covers offset calibration only.

Offset calibration consists of calculating the offset error for the sensor in the finished module and storing this value in memory. The stored value can be subtracted from any future measurements the final application performs. This document describes a possible implementation of sensor offset calibration and measurement correction.

2 Preliminary information for sensor calibration

The TPMS calculates the sensor offset error, also referred to as the calibration coefficient, by comparing the value measured by the sensor with a reference value. The offset error is stored in memory and used to correct the subsequent measurements by subtracting the stored offset error value from future measurements.

2.1 Preparing the reference value

Place the finished module in an environment where the parameter of interest (acceleration, pressure, temperature, or voltage) has a known, precisely measured value. This value is used as the reference value for the sensor.

Example: To perform X-axis acceleration calibration at 0 g, place the TPMS module in a position so that the X-axis accelerometer sees 0 g exactly and is perfectly perpendicular to the gravity force direction.

This reference value can be:

- A fixed predefined value, known in advance by the TPMS, i.e., defined in the calibration function. For example, in case of 0 g acceleration calibration, the TPMS code can define the reference acceleration to be 0 g. The equipment positioning the module must ensure that the accelerometer is in the 0 g position exactly. Using a fixed, predefined value, all modules perform the calibration using this value.

- Communicated to the TPMS at the last moment, via LF for example. Use this method when you cannot ensure that the parameter of interest will always exactly equal a pre-defined value. For example, in the case of temperature calibration at 60 °C, it may be difficult to ensure that all TPMS sensors register at exactly 60 °C. The actual temperature of the sensor may equal to 59 °C or 61 °C. Instead, the actual temperature, in which the TPMS is placed, is measured with precision equipment, and the value communicated to the TPMS.

The precision of the reference value measurement is a key point that directly impacts the result of the correction: if the reference value is not measured accurately, then the offset calculated by the TPMS may include an error, and the correction made on subsequent measurements will include the same error.

2.2 Validity range of the calibration

The offset error is not constant over the entire sensor range, therefore one unique calibration coefficient cannot be used for the whole sensor range and is applicable to all sensors, i.e., acceleration, pressure, temperature and voltage. The goal of the calibration process, described in this document, is to reduce the sensor tolerance around a specific point. To reduce the offset error along several points of the range, a calibration coefficient must be calculated at each point.

Note: *Regarding acceleration calibration, the whole range of acceleration is divided into 16 offset steps which overlap, i.e., the same acceleration can be measured at several offset steps. Each of the 16 offset steps has a slightly unique transfer function versus the ideal transfer function due to the electronic signal chain variances at each step. This means the offset error value varies from one offset step to another, and the calibration performed at a given step is valid at this step only.*

Example: For a more precise determination of car movement, acceleration calibration is usually performed to reduce the offset error at 0 g. For the X-axis, 0 g is in the middle of offset step 7; the X-axis, 0 g calibration must be done at offset step 7 and is valid *only* at offset step 7. The offset error calculated at 0 g at offset step 7 cannot be used for offset step 6, even if 0 g can also be measured at offset step 6. The 0 g offset error at offset step 6 is different from the 0 g offset error at offset step 7.

Calibration coefficient validity around one point implies that, prior to applying correction to the measurements, the application must first check that the measurement is within the calibration validity range, around the point of interest. Measurements not within the calibration range, cannot be corrected with the calibration coefficient taken within the calibration range since the offset error varies along the whole sensor range.

Note: *The calibration coefficient remains valid over the device lifetime as the variation of the offset error with time can be considered negligible compared to other sources of error.*

3 Calculating the offset error coefficient

3.1 Flow

Place the TPMS module in the environment:

- with the appropriate temperature or pressure for temperature or pressure calibration or
- in the right position for acceleration calibration or
- supplied with the appropriate voltage for voltage calibration.

The TPMS should be notified (via LF for example) that the offset determination can be performed. At this moment, the TPMS performs a compensated sensor measurement and compares the value with the reference value, either defined in the code or previously communicated. The difference between the measurement and the reference value is the offset error. This offset error is stored in memory and to be used to correct future measurements taken by the application. The correction process is described in Section 4 "Correcting the sensor measurements".

Note the offset error must be calculated on compensated measurements, not on raw measurements. The TPMS must read and compensate the sensor measurement and use the compensated measurement to calculate the offset error. In addition, the reference value must be converted from common units (g, kPa, °C or Volt) to counts. The conversation formula is provided in the data sheet and varies from part number to part number. Refer to the appropriate data sheet.

The function performing temperature calibration could have the following prototype:

```
Calib_Status Calibrate_Sensor (UINT16 u16ref_value);
```

Description: The function calculates the sensor offset error and stores it in FLASH memory.

Input: u16ref_value, the reference value in counts.

Output: The status of the calibration, which could take the following values (*enum* type):

- "*success*" when the calibration could be successfully performed;
- "*not executed*" when the calibration could not be completed;
- "*sensor error*" when the sensor measurement could not be read or compensated successfully;
- "*write error*" when the offset error coefficient could not be written in FLASH successfully.

The function could perform the following actions as shown in Figure 1.

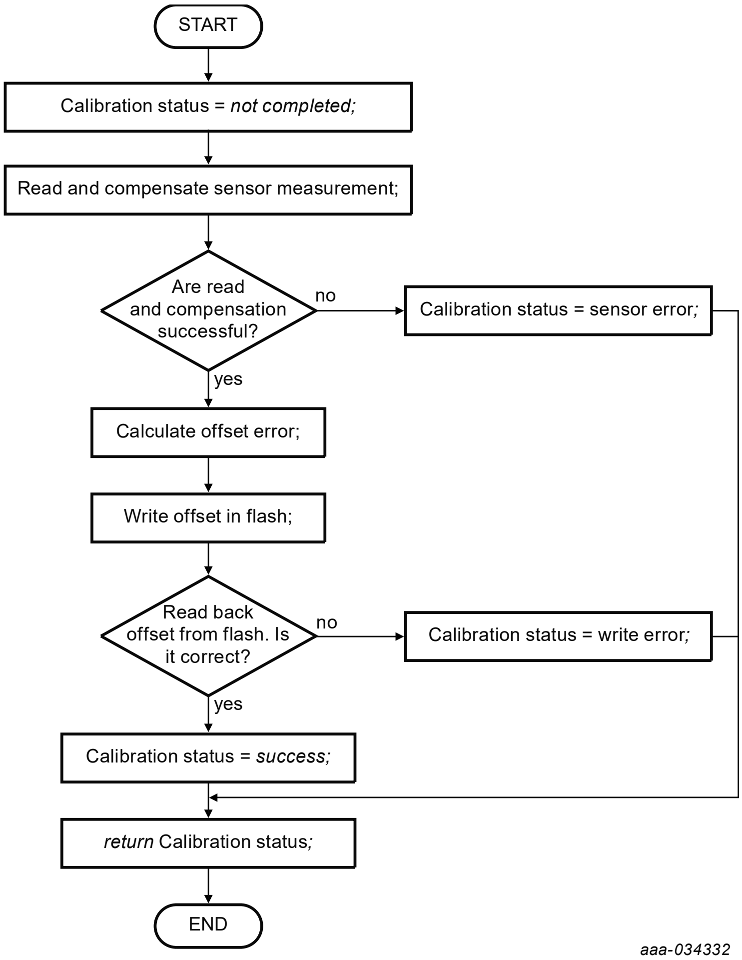

Figure 1. Calibration function flow

- Recommendations for sensor reading and compensation:
  - Configure STOP4 mode before executing any sensor reading.
  - Enable the band gap and execute a 25 µs settling delay before executing temperature or voltage reading.
  - Acceleration and pressure compensations require temperature and voltage raw measurements.
  - Refer to the firmware user guide for more information on the status flags returned by the functions. The status flags allow the application determine if the reading(s) and compensation were successful.
- The offset error can be calculated with the following formula:

  Offset error = compensated measurement – reference value;

  The *compensated measurement* is calculated by the TPMS in the calibration function. The *reference value* is the actual compensated value, in counts, passed as an input parameter to the function. If the sensor has no deviation, the compensated measurement equals the reference value and the offset error equals 0.
- The offset error coefficient can be stored in FLASH memory with the function TPMS_FLASH_WRITE.

  As additional verification, read the coefficient from FLASH and if it contains the expected value, the write sequence was successful. An example is provided here:

```
if( (*(INT16*)(COEFF_START_ADDRESS)) == i16offset_error)
{ // write was successful }
```

Where COEFF_START_ADDRESS is the start address of the calibration coefficient location in FLASH (the coefficient is stored on two bytes).
An unsuccessful write sequence likely indicates that the FLASH location, where the coefficient was written to, was not empty (the bits were not all set to logic 1). See Section 3.3 "Declaring the calibration coefficient" for more details.

3.2 Types of the variables

Compensated sensor measurements are unsigned values from 8 to 12 bits, depending on the sensor and TPMS part number. The offset error is either positive or negative and must be stored as a signed value with one bit for the sign (the MSBit) plus 8 to 12 bits for the value. This implies the calibration coefficient must be a signed 16-bit value.

With sensor compensated values as unsigned values, perform appropriate type casting during the calculations to avoid any type issue:

1. Compensated measurements and reference values originally stored in 8-bit values (temperature and voltage, for example) should first be cast into unsigned 16-bit values.
   Example: u8sensor_meas is the unsigned 8_bit sensor measurement, and *u16sensor_meas* is its 16-bit representation. The following cast can be done:
   ```
   u16sensor_meas=(UINT16)(u8sensor_meas);
   ```
2. During offset calculation, the unsigned sensor values may be cast into signed sensor values to ensure that a signed value is stored in the calibration coefficient. The cast from unsigned to signed does not impact the sensor values as the MSBit is always clear and sensor values are always represented as positive values shorter than 16 bits. The following cast can be done:
   ```
   i16offset_error=(INT16)(u16sensor_meas)-(INT16)(u16ref_value);
   ```

3.3 Declaring the calibration coefficient

Declare the calibration coefficient containing the offset error as a signed 16-bit constant, initialized to 0xFFFF. Initialize the coefficient with logic 1, not logic 0. Only logic 0 bits can be written in FLASH and the function TPMS_FLASH_WRITE can only write logic 0 bits in FLASH. To set a FLASH bit to logic1 requires erasing FLASH, which is done only page per page, so 512 bytes per 512 bytes.

An example declaration is provided here:

```
const INT16 OFFSET_ERROR @COEFF_START_ADDRESS=0xFFFF;
```

Carefully choose the address of the coefficient to ensure that the address is not located in a FLASH location that may be overwritten by other application functions, as in the case of "Over The Air reprogramming".

4 Correcting the sensor measurements

Once the calibration has been performed, compensated sensor measurements can be corrected, i.e., the offset error can be subtracted from sensor measurements, provided they are around the calibration point.

The correction function could have the following prototype:

UINT16 **Correct_sensor_meas** (UINT16 u16sensor_meas);

Description: the function checks that a calibration coefficient has been stored in memory, that we are in the correction validity range and if so, applies correction to the sensor measurement value given as parameter.

Input: the compensated sensor measurement in counts that was calculated by the device.

Output: the corrected compensated measurement, in counts.

Note: *In this example, the non-corrected measurement is passed as parameter, and the corrected measurement is returned. This can be useful for debug purpose during development, for the user to see the difference between corrected and non-corrected values. But it is also possible to write the function such that it takes as parameter the pointer to the variable storing the sensor measurement and overwrites it with the corrected measurement.*

The function could have the flow presented in Figure 2.

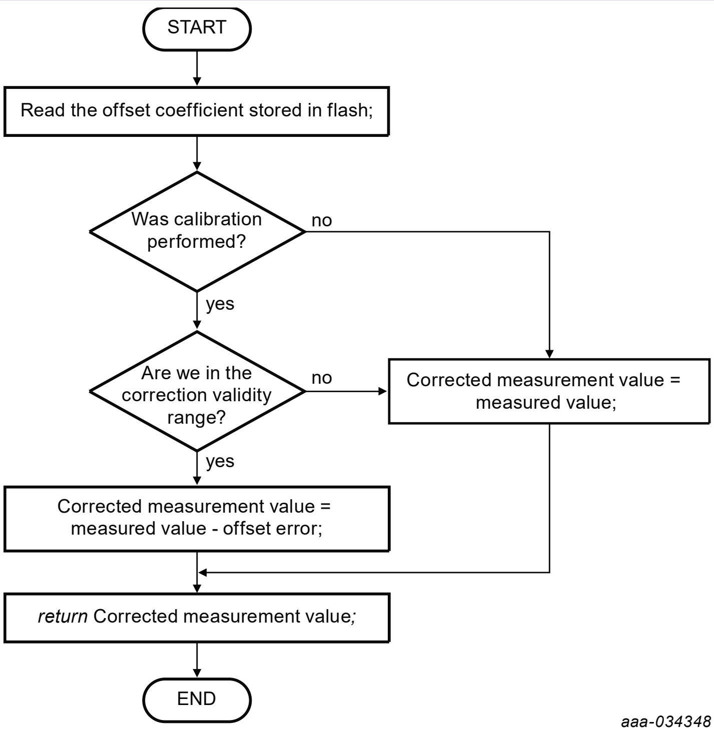

Figure 2. Correction function flow

- First check that the calibration coefficient has been stored in FLASH. If the FLASH location is empty, i.e., equal to the initialization value 0xFFFF, it means the calibration was not executed or was not successful and the correction cannot be performed. In this case, the sensor measurement is not modified.

```
i16offset_error=*(INT16*)(COEFF_START_ADDRESS);
if(0xFFFF!=i16offset_error)
{
  // calibration was performed, continue…
}
```

Where COEFF_START_ADDRESS is the location start address of the calibration coefficient in FLASH (the coefficient is stored on two bytes).

- As previously explained, the calibration coefficient is only valid around the calibration point. Before applying correction, verify the calibration coefficient is within the correction validity range around the calibration point. The following can be implemented.

```
if((
    (u16sensor_meas-i16offset_error) >=
      CORRECTION_RANGE_THRESH_LOW)
    && ((u16sensor_meas - i16offset_error) <
      CORRECTION_RANGE_THRESH_HIGH))
{
  // we are in the validity range, apply correction…
}
```

  Where CORRECTION_RANGE_THRESH_LOW and CORRECTION_RANGE_THRESH_HIGH are the limits of the correction range, in counts, defined around the calibration point.
  Within the correction range, apply the correction. Subtract the offset error from the compensated sensor measurement passed as a parameter and the corrected measurement is returned by the function.
  In the offset subtraction calculation, pay attention to the types of variables. Since the offset value is a signed value, a cast should be performed to convert the unsigned 16-bit sensor measurement into a signed value. The signed result can then be converted back to an unsigned 16-bit value. The successive casts do not modify the sensor values as they always remain positive values shorter than 16 bits and the MSBit remains clear at all steps.

```
u16corrected_meas=
  (UINT16)((INT16)u16sensor_meas - i16offset_error);
```

  For 8-bit sensor values, the corrected value can then be stored in an unsigned 8-bit value.
- In the event correction cannot be performed, the corrected measurement is set to equal the sensor measurement passed as a parameter.

5 Revision history

Table 1. Revision history

Revision	Date	Description
1	20190701	Initial release

6 Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Suitability for use in automotive applications — This NXP Semiconductors product has been qualified for use in automotive applications. Unless otherwise agreed in writing, the product is not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — While NXP Semiconductors has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP Semiconductors accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

6.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

NXP — is a trademark of NXP B.V.

Contents

Date of release: 1 July 2019
Document identifier: AN12519