AN13371

TPMS Clocks Calibration

Rev. 1 — 1 October 2021 Application note

Document information

Information	Content
Keywords	FXTH, NTM88, clocks, calibration, oscillator drift, temperature
Abstract	FXTH and NTM88 devices include three internal oscillators, a low frequency, medium frequency, and high frequency oscillator. The oscillators drift with temperature which may impact user applications that rely on precise timing. This application note documents how to compensate for the drift.

Revision history

Rev	Date	Description
1	20211001	• Initial release

1 Introduction

The FXTH and NTM88 devices include three internal oscillators: the Low Frequency Oscillator (LFO) which has a target frequency of 1 kHz, the Medium Frequency Oscillator (MFO) which has a target frequency of 125 kHz, and the High Frequency Oscillator (HFO) which is derived from the MFO and has a target frequency configurable by the user of either 1 MHz, 2 MHz, 4 MHz or 8 MHz. The bus clock is derived from the HFO and has a frequency equal to half the frequency of the HFO, so either 0.5 MHz, 1 MHz, 2 MHz or 4 MHz.

These oscillators drift with temperature, which can impact the user application in case it relies on precise timings. This application note explains how to compensate for the drift. The first part focuses on the LFO and the second part on the Bus Clock.

2 Calibrating the PWU to compensate for the LFO drift

2.1 Purpose

In TPMS devices, the Low Frequency Oscillator (LFO) has a targeted frequency of 1 kHz and clocks several blocks. Among them is the Periodic Wake-Up (PWU) block.

The actual frequency of the LFO varies from one device to another and with temperature. The frequency deviation is specified in the data sheet of the product[1].

A possible consequence of LFO frequency drift is a higher power consumption. To illustrate, we can take the example of an application in which the PWU is configured to wake-up the TPMS device from STOP1 every 30 seconds to send the tire pressure value. If the LFO has an actual frequency of 1.2 kHz instead of 1 kHz, then the PWU wakes up the TPMS approximately 20 % more often than expected. The additional wake-ups result in a higher power consumption, and shorter battery life.

On the contrary, if the actual LFO frequency is lower than 1 kHz the TPMS will wake-up less often than expected: instead of sending a frame every 30 seconds, the TPMS could send a frame every 40 seconds, for example, which may not meet the application requirements.

The PWU period is configured with the WDIV value in the PWUDIV register. Refer to the user manuals[1],[2] for more information. The purpose of the WDIV calibration described in this section is to calculate a WDIV value based on the actual LFO frequency. Note the WDIV calibration does not modify the LFO frequency.

2.2 Principle of WDIV calibration

The Periodic Wake-Up block is clocked by the Low Frequency Oscillator. The PWU can be configured to generate a periodic wake-up and/or a periodic reset. The wake-up and reset periods are configured in two or three steps:

1. The WDIV field of PWUDIV register configures the prescaler for the incoming LFO clock period. The output period is **WCLK**.
2. For periodic wake-up and/or periodic reset: the PWUCS0_WUT field configures the number of WCLK clocks before the next periodic wake-up interrupt is generated. The output period is **RCLK**.

1 Contact an NXP representative to have access to the data sheets.

3. For periodic reset only: the PWUCS1_PRST field configures the number of RCLK clocks before the next periodic reset is generated.

When the LFO frequency is exactly equal to 1 kHz, configuring WDIV = 31 results in a 1 second output period for WCLK. In this case, if the user wants to configure a periodic wake-up of 30 seconds and a periodic reset of 60 seconds, WUT field must be set to 30 and PRST field must be set to 2.

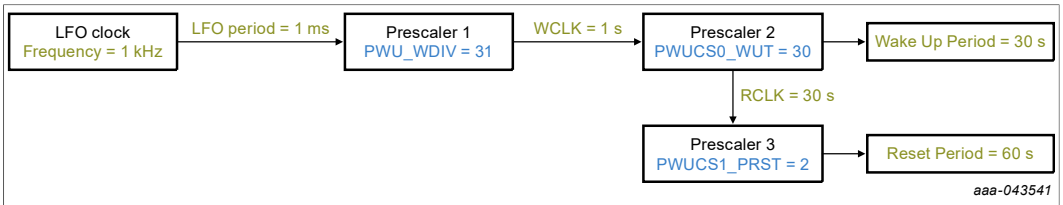

Figure 1. Wake-up and Reset period configuration when LFO frequency is 1 kHz

However, the LFO frequency is not always equal to 1 kHz: it varies from one device to another, and for one given device the LFO frequency drifts with temperature meaning that the LFO frequency varies when temperature varies.

When the actual LFO frequency deviates from 1 kHz, this implies that configuring WDIV to 31 does not result in an exact 1 second output for WCLK: for example, if the actual LFO frequency is equal to 1.2 kHz, the resulting WCLK period will be 833 ms. In this case, if WUT is set to 30, this results in a 25 second periodic wake-up, instead of the 30 second periodic wake-up expected by the user.

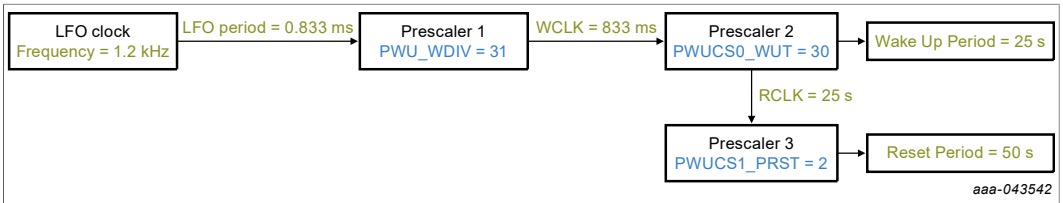

Figure 2. Wake-up and Reset period configuration when LFO frequency is 1.2 kHz

The purpose of the LFOCAL functions performing WDIV calibration is to have **WCLK = 1 second**.

For that, the functions calculate the actual LFO frequency and return a calibrated value of WDIV i.e. a value of WDIV which results in WCLK = 1 second. For example, if the actual LFO frequency is 1.2 kHz, the LFOCAL function will return value 43, meaning that WDIV must be set to 43 instead of 31 to have WCLK = 1 second. If the actual LFO frequency is 800 Hz, the LFOCAL function returns the value 19.

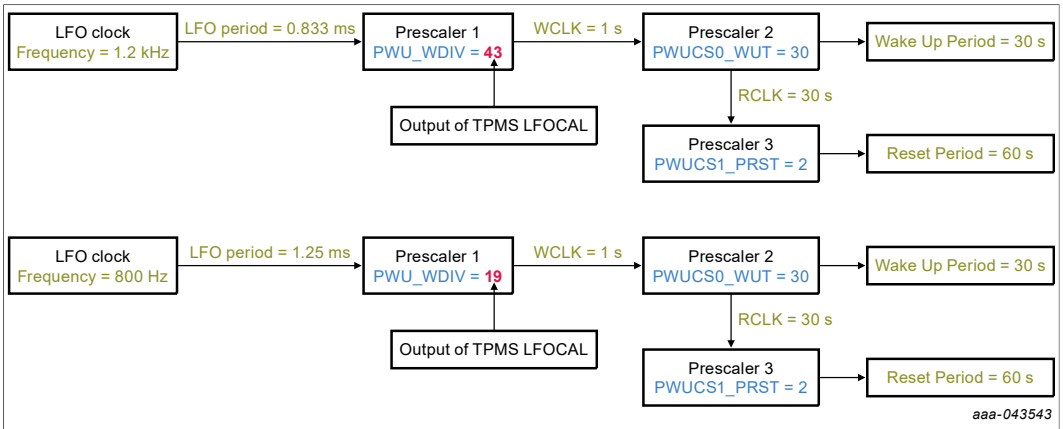

Figure 3. Wake-up and Reset period configuration with WDIV calibrated

2.3 Calculation of the LFO actual frequency with 26 MHz crystal or bus clock

To return a calibrated value of WDIV, it is necessary to calculate the actual frequency of the LFO. In the example above, this means we need to know that the actual LFO frequency is 800 Hz to be able to calculate that WDIV must be set to 19.

To calculate the actual LFO frequency, we use a reference clock, meaning a clock more precise than the LFO. The functions developed to perform WDIV calibration use either the external 26 MHz crystal or the bus clock. In these functions, TPM1 timer is used to count the number of RF crystal or bus clocks periods per LFO period; this allows for the calculation of the actual frequency of the LFO and then calculate a calibrated WDIV value.

The 26 MHz RF crystal is the most precise clock that can be used to calculate the LFO frequency: the frequency deviation of the RF crystal on the whole temperature range is negligible, much smaller than the frequency deviation of any internal clock of the TPMS, including the bus clock. However, note that using the 26 MHz crystal to measure the LFO frequency requires the RF block to be enabled when the LFOCAL function is executed, which consumes additional power. The extra current coming from the use of the crystal can be estimated to be equivalent to the sum of the *RF transmission current at 315 MHz, 5 dBm* during ~750 µs and the *Interframe period current with IFPD=0* during the rest of the function execution time.

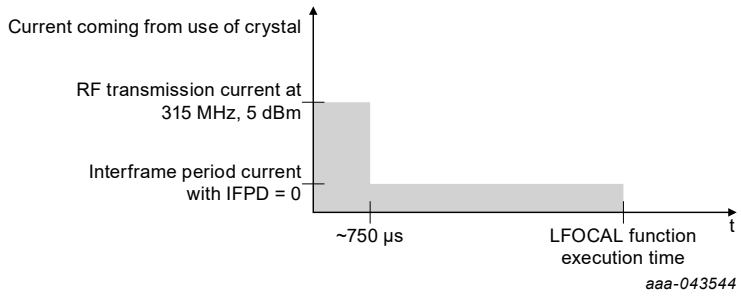

Figure 4. Equivalent of the current consumed during the execution of the LFOCAL function coming from the use of the 26 MHz crystal

The *RF transmission current at 315 MHz, 5 dBm,* and the *Interframe period current with IFPD=0* are specified in the data sheet[2] of the product.

For applications in which an external 26 MHz crystal is not present, or in case the user wants to avoid the extra current consumption brought by the use of the crystal, the bus clock can be used as reference clock. However, unlike the RF crystal, it drifts with temperature – but its drift remains much smaller than the LFO drift. The range of bus clock frequency is specified in the data sheet of the product[2].

2.4 Functions performing WDIV calibration

2.4.1 For the FXTH

The function TPMS_LFOCAL is available in the FXTH firmware and firmware library and is described in the FXTH Firmware User Guide[3]. This function uses the 26 MHz external crystal to calculate the LFO frequency. It returns valid results when the LFO frequency range remains between 660 Hz and 1900 Hz, which is the case for FXTH87 devices but not necessarily for devices of the FXTH87E family. As a consequence, this function is not recommended for FXTH87E devices.

The library “New TPMS LFOCAL” that can be downloaded from the TPMS Software webpage[5] includes functions compatible with both FXTH87 and FXTH87E devices, which use either the 26 MHz external crystal or the bus clock as reference clock. These functions are described in the user guide provided with the library.

2.4.2 For the NTM88

Two functions are available in the firmware library: TPMS_LFOCAL, which uses the 26 MHz crystal as reference clock, and TPMS_LFOCAL_BUSCLK, which uses the bus clock. Both functions are described in the NTM88 Firmware User Guide[6].

2.5 Implementing WDIV calibration in the application

As explained in the previous section, the LFOCAL functions return a calibrated WDIV value. The functions measure the LFO frequency and return a WDIV value to have WCLK equal to 1 second. The WDIV compensated value depends on the LFO frequency. This implies that the WDIV value must be re-calculated when the LFO frequency varies. For example, if the WDIV value is first calculated when the LFO frequency is equal to 800 Hz the LFOCAL function will return a WDIV value of 19; but if the LFO frequency varies after some time and becomes equal to 700 Hz, this WDIV value of 19 will not be accurate anymore, it needs to be re-calculated: the LFOCAL function needs to be executed again and will return the value 12 which must be used instead of 19.

- *How to know when the LFO frequency changes and the compensation function needs to be executed again?*

The LFO frequency drifts with temperature so the WDIV compensated value needs to be recalculated when temperature changes. It is recommended to execute the LFOCAL function when temperature varies by 20 °C. A possible implementation for the application is described below.

2 Contact an NXP representative to have access to the data sheets.

Inside the application code, an array of 8 bytes UINT8 WDIV[8] is declared in the PARAM section maintained in STOP1. The whole device temperature range from -40 °C to 125 °C is divided into 8 ranges of 20 °C as shown in Figure 5. Each byte of the WDIV[] array represents the WDIV compensated value on a given temperature range.

For example, WDIV[3] contains the compensated WDIV value when temperature is between 20 °C and 40 °C. In the application, when the temperature is between 20 °C and 40 °C for the first time the LFOCAL function is executed and the WDIV value is stored in WDIV[3]. Then each time the current temperature is again between 20 °C and 40 °C the compensated value stored in WDIV[3] can be used, without needing to execute the LFOCAL function again.

This means the LFOCAL function will be executed a maximum of eight times in all the device lifetime, corresponding to once per temperature range. Once the WDIV value has been calculated for a temperature range, the result is stored in WDIV[] array and there is no need to execute the LFOCAL function again for this temperature range.

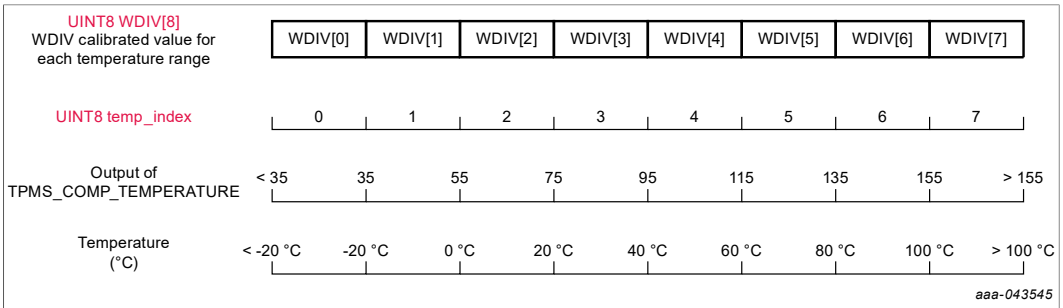

Figure 5. Correspondence between the index of WDIV[8] array and temperature

An example flow is shown in Figure 6.

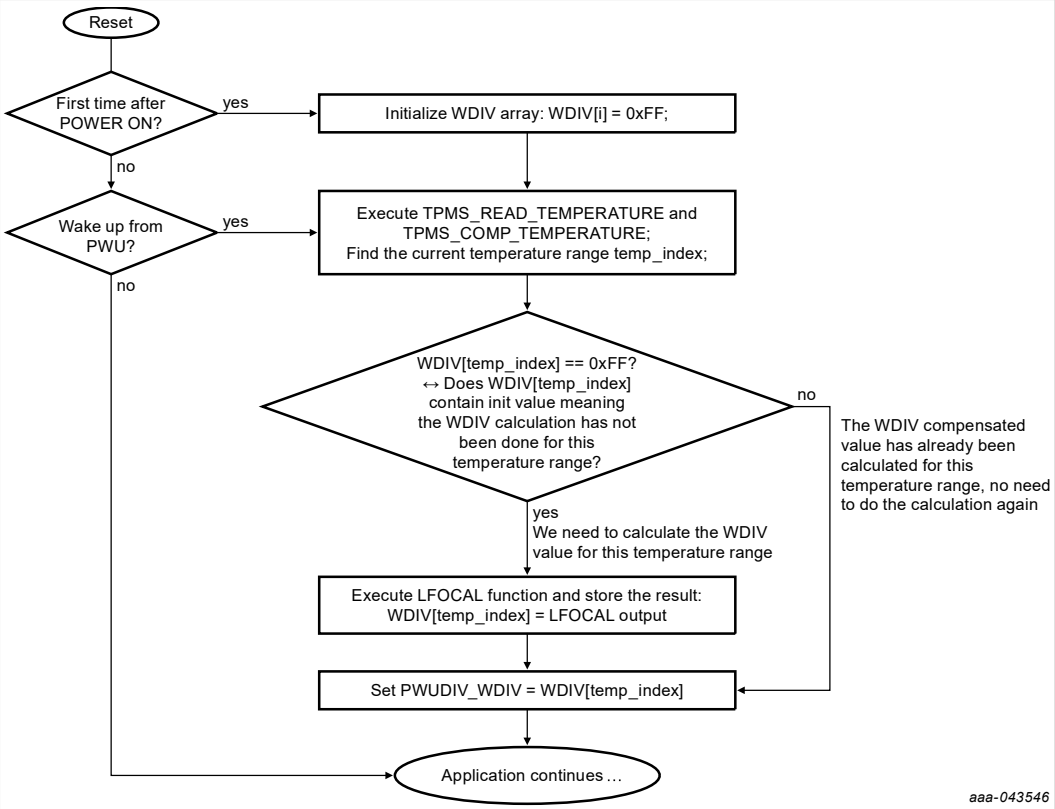

Figure 6. Example of WDIV calibration implementation in the application

The TPMS projects available in the FXTH or NTM88 Starter Package implement the algorithm above. Download the Starter Packages from the TPMS Software webpage[5] to see an example of source code.

Note: *An alternate implementation would be to execute the LFOCAL function at each PWU wake-up without checking the current temperature. In this case, the calibrated WDIV value would be calculated again at each PWU wake-up. The drawback of this alternate implementation is that it would consume more power than executing the LFOCAL function only when temperature changes because it consumes more power to execute the LFOCAL function at every wake-up than performing a temperature acquisition and compensation at every wake-up and performing the LFOCAL function a maximum of eight times in the device lifetime.*

3 Calibrating the Bus Clock frequency

3.1 Purpose

The actual frequency of the bus clock varies from one device to another and with temperature. The frequency range is specified in the data sheet of the product[3].

The execution time of the instructions performed by the program directly depend on the bus clock frequency. Also, the bus clock can be configured as clock source for TPM1 timer, which can be used to perform delays in the application. This implies that

3 Contact an NXP representative to have access to the data sheets.

applications that require precise timings may need to calibrate the bus clock frequency to ensure a maximum timing accuracy.

3.2 Principle of bus clock calibration

It is possible to measure the actual frequency of the bus clock using the 26 MHz external crystal as reference. However, it is not possible to adjust the bus clock frequency directly, only the MFO frequency can be modified.

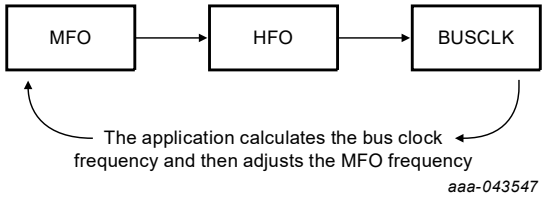

Figure 7. Principle of bus clock calibration

The firmware function TPMS_MFOCAL measures the actual bus clock frequency and returns a value indicating the percentage of deviation compared to the target frequency. This function is available in the FXTH firmware and firmware library and the NTM88 firmware library, and described in the firmware user guides [3],[6]. Note that in the description of the function it is indicated that the percentage of deviation of the MFO frequency is returned, but it is actually the percentage of deviation of the bus clock frequency. When TPMS_MFOCAL function returns value 128 it means that the frequency of the bus clock is the target frequency. Each LSB away from this value corresponds to a deviation of 0.78 %; when the frequency is close to the target frequency this corresponds to approximately 975 Hz.

Register SIMOTRM[7:0] allows for the modification of the MFO frequency. At production, NXP trims this register so that the bus clock frequency is as close as possible to its target frequency at 29 °C. Increasing or decreasing the register value by one count increases or decreases the MFO frequency by approximately 250 Hz. The application can update this register at any time, and the new value will be maintained until a reset occurs. At reset, the value trimmed by NXP at production is loaded into SIMOTRM register, overwriting the value that the application configured.

The MFO, HFO, and bus clock frequencies drift with temperature, which implies that a different SIMOTRM value needs to be calculated when temperature varies.

3.3 Implementing bus clock calibration in the application

3.3.1 Adjusting the bus clock frequency

The principle of adjusting the bus clock frequency is the following: the application uses the MFOCAL function to calculate the bus clock deviation with a ~975 Hz precision, and then adjusts the MFO frequency via SIMOTRM register with a ~250 Hz resolution. It may take several iterations to reach the target frequency, which would imply executing the MFOCAL function potentially several times. Since doing so would consume more power, the user could configure the maximum number of times MFOCAL function can be executed in the algorithm. In the example of implementation described below, the variable *Nb_executions* keeps track of how many times the MFOCAL function was executed and the value MAX_EXE_NB represents the maximum number of times the TPMS_MFOCAL function can be executed.

An example of implementation is detailed in the flow below:

- When the bus clock frequency is calculated to be equal to the target frequency, MFOCAL function returns 128. So, the drift is calculated as the difference between the value returned by MFOCAL function and value 128. The drift is positive when the bus clock frequency is higher than the target frequency, and negative otherwise.
- SIMOTRM register is first adjusted by a number of counts equal to four times the drift value, since 975 Hz ~= 4*250 Hz (refer to the paragraph above).
- After that, and as long as the drift is not 0 and the MFOCAL function has not been executed more than the max number of times fixed by the user, the new drift is calculated and SIMOTRM register adjusted in consequence, but by smaller steps, count by count.
- A minimum bus clock settling delay of 500 µs is necessary between the moment MFO frequency is adjusted and the moment the bus clock frequency is measured.

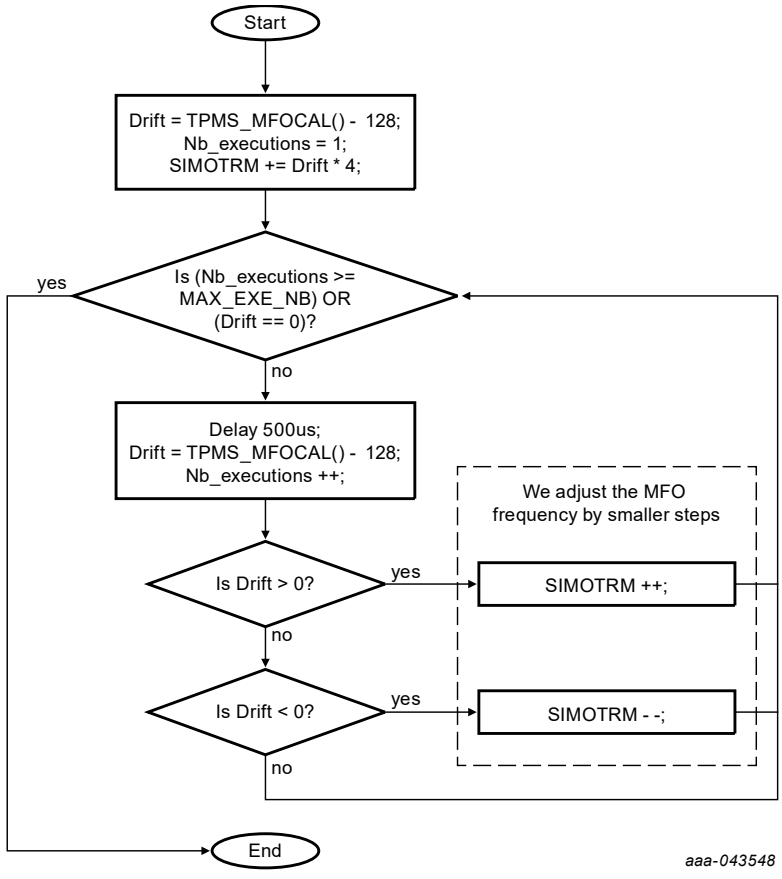

Figure 8. Example of flow to adjust SIMOTRM

The corresponding source code is the following:

```
/* MFOCAL execution */
#define MAX_EXE_NB 3
void vfnCalibSIMOTRM (void)
{
UINT8 u8MfocalValue, u8NbExecutions;
INT8 i8Drift;
 TPMS_RF_ENABLE(1);
 u8MfocalValue = TPMS_MFOCAL();
 u8NbExecutions = 1; // We have executed MFOCAL once
```

```
if (u8MfocalValue != 255) // 255 is error code for crystal not
 present
 {
i8Drift = u8MfocalValue - 128;
 SIMOTRM += (i8Drift * 4);
while ((u8NbExecutions < MAX_EXE_NB) && (i8Drift != 0) &&
 (u8MfocalValue != 255))
 {
 Delay_us(100); // 500 µs delay to let time to bus clock
 frequency to stabilize
u8MfocalValue = TPMS_MFOCAL();
u8NbExecutions ++; // We have executed MFOCAL once more
i8Drift = u8MfocalValue - 128;
if ((u8MfocalValue != 255) && (i8Drift > 0))
SIMOTRM ++;
else if ((u8MfocalValue != 255) && (i8Drift < 0))
SIMOTRM --;
 }
 }
 TPMS_RF_ENABLE(0);
return;
}
//
=====================================================================
// Delay in us:
// us4 delay
// 0 10.6  µs
// 1 15.0 µs
// 2 19.4 µs
// 3 23.8 µs
// 4 28.2 µs
// 10 53.4 µs
// 20 96.2 µs
// 255 1099 µs
// delay = 10.6 µs + (us4 * 4.28 µs)
// us4 = (delay-10.6 µs)/4.28 µs
void Delay_us(UINT8 us4)
{
UINT8 u8;
for (u8=0; u8<us4; ++u8)
{
__asm NOP;
__asm NOP;
__asm NOP;
__asm NOP;
 }
return;
}
```

3.3.2 Storing SIMOTRM value for each temperature range

The bus clock frequency changes with temperature, so it is necessary to re-calculate SIMOTRM value when temperature changes. It is recommended to re-calculate the value when temperature varies by 20 °C. A possible implementation for the application is described below.

Inside the application code, an array of 8 bytes UINT8 gau8SIMOTRM[8] is declared in the PARAM section maintained in STOP1. The whole device temperature range from -40 °C to 125 °C is divided into 8 ranges of 20 °C as shown in Figure 9. Each byte

of the gau8SIMOTRM[] array represents the calibrated SIMOTRM value on a given temperature range.

For example, gau8SIMOTRM[3] contains the calibrated SIMOTRM value when temperature is between 20 °C and 40 °C. In the application, when the temperature is between 20 °C and 40 °C for the first time, SIMOTRM value is calculated and stored in gau8SIMOTRM[3]. Then each time the current temperature is again between 20 °C and 40 °C the calibrated value of SIMOTRM stored in gau8SIMOTRM[3] can be used, without needing to re-calculate it again.

This means the calculation of SIMOTRM will be executed a maximum of eight times in all the device lifetime, corresponding to once per temperature range. Once the SIMOTRM value has been calculated for a temperature range, it is stored in gau8SIMOTRM[] array and there is no need to calculate it again for this temperature range.

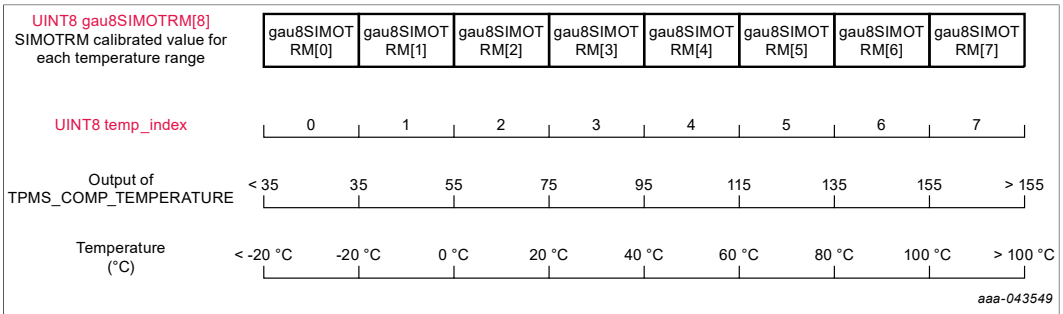

Figure 9. Correspondence between the index of gau8SIMOTRM[8] array and temperature

An example of flow is shown in Figure 10.

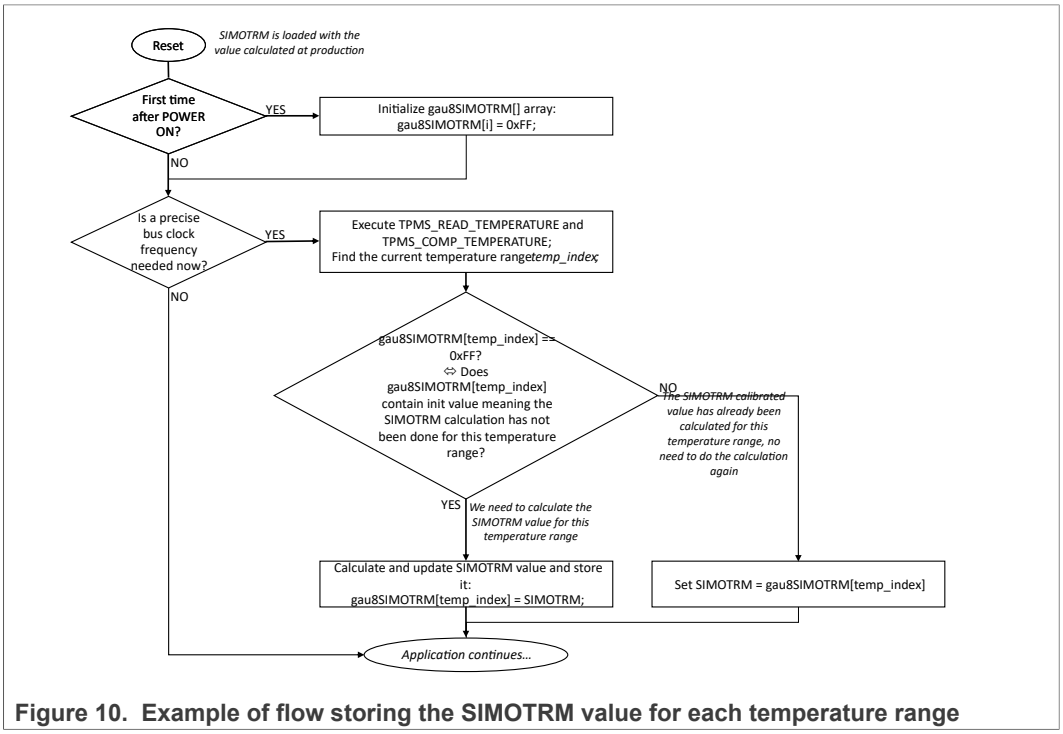

Figure 10. Example of flow storing the SIMOTRM value for each temperature range

The source code corresponding to the *temp_index* search and adjustment of SIMOTRM value is the following:

```
#define DEFAULT_RANGE_INDEX 3
#define LIMIT_LOWEST_RANGE 35
#define RANGE_LENGTH 20
#define SIMOTRM_MAX_INDEX 7
#define SIMOTRM_INIT_VALUE 0xFF
/*************************************************************
 * This functions checks if SIMOTRM needs to be re-calculated
 * or not, depending on whether its value has been calculated
 * for the currenttemperature range.
 * If it has, SIMOTRM is updated with the stored value.
 * If not, it is calculated and its value is stored.
 * PRE-REQUISITE: valid compensated temperature measurement
 * must be stored in gu8CompTemp.
 *************************************************************/
void Calibrate_SIMOTRM_with_Temperature (void)
{
UINT8 temp_index;
UINT8 gu8CompTemp_temporary;
/* Temperature value will be modified, store it in temporary
 variable */
gu8CompTemp_temporary = gu8CompTemp;
/* Find current temperature range in the gau8SIMOTRM array */
temp_index = 0;
while ((gu8CompTemp_temporary > LIMIT_LOWEST_RANGE) &&
 (temp_index < SIMOTRM_MAX_INDEX))
{
gu8CompTemp_temporary -= RANGE_LENGTH;
temp_index ++;
}
/* Has calibration already been done for this range? */
if (gau8SIMOTRM[temp_index] == SIMOTRM_INIT_VALUE)
{
/* Compensation has not yet been done, do it now */
vfnCalibSIMOTRM();
gau8SIMOTRM[temp_index] = SIMOTRM;
}
else
{
/* Update SIMOTRM with the value stored */
SIMOTRM = gau8SIMOTRM[temp_index];
}
return;
}
```

Note: *An alternate implementation would be to re-calculate the SIMOTRM value each time the application needs a precise bus clock frequency without checking the current temperature. The drawback of this alternate implementation is that it would consume more power than calculating SIMOTRM only when temperature changes because it consumes more power to calculate SIMOTRM value at every wake-up than performing a temperature acquisition and compensation at every wake-up and performing the SIMOTRM calculation a maximum of eight times in the device lifetime.*

3.4 Note on using the 26 MHz crystal to perform both WDIV and bus clock calibrations

Performing WDIV calibration using the 26 MHz crystal as reference and performing bus clock calibration both require the RF block to be enabled when the calibrations are performed. The RF block can be enabled by calling the function TPMS_RF_ENABLE(1), which turns on the block and then performs a ~300 µs settling delay. This function is described in the firmware user guides[3],[6]. In applications using the crystal for both WDIV and bus clock calibration, the two calibrations can be performed one after the other, without disabling the RF block in between. In other words, the application can enable the RF block, perform both calibrations, and then disable the RF block. This allows a bit of power saving by not disabling and enabling again the RF block between the two calibrations.

4 References

[1] **FXTH87E** — *FXTH87E, Family of Tire Pressure Monitor Sensors*, reference manual
https://www.nxp.com/docs/en/reference-manual/FXTH87ERM.pdf

[2] **UM11227** — *NTM88 family of tire pressure monitor sensors*, user manual
https://www.nxp.com/docs/en/user-guide/UM11227.pdf

[3] **FXTH87xx02FWUG** — *FXTH87xx02 Embedded Firmware User Guide*, user guide
https://www.nxp.com/docs/en/user-guide/FXTH87xx02FWUG.pdf

[4] **FXTH87xx1xFWUG** — *FXTH87xx11 and FXTH87xx12 embedded firmware user guide*, user guide
https://www.nxp.com/docs/en/user-guide/FXTH87xx1XFWUG.pdf

[5] **TPMS Software Webpage** —
https://www.nxp.com/design/sensor-developer-resources/tpms-software:TPMS-SOFTWARE

[6] **UM11145** — *NTM88 Firmware Library User Guide*, user guide
https://www.nxp.com/docs/en/user-guide/UM11145.pdf

5 Legal information

5.1 Definitions

Draft — A draft status on a document indicates that 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 in a draft version of a document and shall have no liability for the consequences of use of such information.

5.2 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.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

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.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

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 — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

5.3 Trademarks

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

NXP — wordmark and logo are trademarks of NXP B.V.

Figures

Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 1 October 2021
Document identifier: AN13371