# AN13882

## Host application for NBP devices

Rev. 2 — 9 August 2023 Application note

**Document information**

| Information | Content |
|---|---|
| Keywords | NBP8, NBP9 |
| Abstract | This application note provides recommendations for the development of the Host application for NBP devices. |

**Revision history**

| Rev | Date | Description |
|---|---|---|
| 2 | 20230809 | • Section 1, removed the content from the introduction and revised the content across three new sections, Section 1.1, Section 1.2, and Section 1.3.<br>• Section 3.1, added an "Important Note" at the end of the section.<br>• Section 9 and Section 10, inserted new sections.<br>• Section 11, added two new references [6] and [7]. |
| 1 | 20230322 | Initial release |

# 1 Introduction

## 1.1 Purpose

This application note provides recommendations for the development of the Host application for NBP devices. The information provided complements the NBP data sheets[2],[3] and Host demo codes[4],[5] available on the NBP product webpage.[1]

Unless otherwise stated, information is applicable to all NBPx part numbers.

## 1.2 Part number selection for battery pressure monitoring applications

Two strategies can be used to monitor the pressure variation in a battery pack. They lead to different implementations at system level:

- The pressure sensor collects pressure measurements and runs algorithms to monitor pressure variation. The sensor performs all processing and notifies the Host only when an event requiring attention occurs. As long as no event occurs, the Host can focus on other tasks, remain in low power, or be turned off to reach very low power consumption.
  The NBP family of pressure sensors was designed for such low-power implementation.
- The pressure sensor takes pressure measurements which are periodically collected by the Host. No advanced processing is performed by the pressure sensor, so the Host needs to run the algorithms to monitor pressure variation.
  The FXPS7250 family of pressure sensors is suitable for such implementation. Two versions exist: the FXPS7250D4S[6] (digital version) with SPI or $I^2C$ interface, and the FXPS7250A4S[7] with analog output (input to Host ADC pin).

Two part numbers are available in the NBP family: the NBP8 and the NBP9. The NBP8 has a SPI interface and the NBP9 has a SPI and PWM interface. For both part numbers, SPI communication with the Host is required in at least the two cases:

- When the application starts, to update the register configuration if the default values do not suit the application.
- When the NBP triggers a notification. SPI communication is required for the Host to acknowledge the notification and clear potential warning or error flags in the NBP.

The Host can also establish SPI communication at any time to read sensor values or update some parameters depending on the state of the battery, like the sampling period for example.

The PWM interface of the NBP9 allows the Host to periodically read the pressure value without having to establish SPI communication. So, the PWM interface allows to reduce the amount of SPI transfers. But it does not replace the SPI interface. SPI communication is still required for the two cases mentioned above. Also, note that the average current consumption of the NBP9 when the PWM signal is generated is higher than the consumption of the NBP8. See Section 10 for more information on power consumption.

Applications in which the Host cannot support SPI communication with the sensor can instead use the FXPS7250A4S[7] analog device or the FXPS7250DI4S[6] $I^2C$ device.

## 1.3 Configuration of the NBP parameters

The purpose of the NBP is to monitor pressure variations to detect thermal runaway events in the battery pack. When a thermal runaway event is ongoing, pressure rises due to the release of gases by the battery cells. If a safety vent is included in the battery pack, pressure rises until the vent opens. Following the opening of the vent, pressure typically starts decreasing. The speed at which pressure rises or decreases and the amplitude of the pressure variation depend on several parameters like the battery energy density, volume of air in the battery

pack, venting threshold etc... The presence or absence of a safety vent will have a direct influence on whether pressure decreases after reaching a certain threshold, or keeps increasing until the battery pack is breached. As a result, the shape of the pressure variation during a thermal runaway event entirely depends on the design of the battery pack.

To support the detection of a wide range of pressure patterns, many parameters in the NBP are user-configurable. Parameters also need to be configured taking into account the target power consumption, noise, etc... All parameters need to be selected based on the specific use-case and requirements of the application.

## 2 NBP program start and default configuration

No action from the Host is required for the NBP embedded application to start running. The NBP application automatically starts after Power-On-Reset (POR). By default, the NBP registers are configured with their reset value indicated in the data sheet[2],[3]. Table 1 details the default configuration. In the calculations, a sensitivity value of 0.206 kPa/LSB is used.

**Table 1. NBP default configuration after POR**

| Register | Name | Reset value | Description |
|---|---|---|---|
| **NBP8:** PSP | Pressure Sampling Period | 0x04 | Sampling period configured to 135 ms.<br>The periodic actions are executed every 135 ms. |
| **NBP9:** MODECFG | Mode Configuration | 0x00 | Eco mode is selected. Sampling period configured to 500 ms.<br>The periodic actions are executed every 500 ms.<br>The PWM signal is asserted at logic 0, idle at logic 1. |
| STPER | Self-test Period | **NBP8:** 0xFF<br>**NBP9:** 0x01 | Periodic self-test period configured to (255 x sampling period) for the NBP8 and (1 x sampling period) for the NBP9.<br>The ADC and Pcell self-test are performed once every 255 sampling periods for the NBP8 and once every sampling period for the NBP9. |
| PINCFG | PIN Configuration | 0x00 | PS ENABLE function is disabled. |
| INTTRIG | Interrupt pulse Trigger | 0x3E | The INT signal is asserted at logic 1, idle at logic 0.<br>The INT pulse has a duration of 8 ms.<br>An INT pulse is generated upon firmware verification error, self-test error, and Sensor error. |
| PCCFG | Pressure Change Configuration | 0x01 | The Fixed Threshold algorithm is enabled. The algorithm is run at every sampling period after a new pressure measurement has been taken. |
| PCDEBT | Pressure Change Debounce Threshold | 0x05 | Debounce Threshold configured to value 5. |
| PCFIXT | Pressure Change Fixed Threshold | 0x0320 | Fixed Threshold configured to value 0x320 = 800 counts. The threshold value in kPa is equal to:<br>*Fixed Threshold (kPa) = 800 x 0.206 + 39.6 = 204 kPa.* |
| PCMINT | Pressure Change Minimum Threshold | 0x03 | Minimum Threshold value configured to 3 counts. The threshold value in kPa is equal to:<br>*Minimum Threshold (kPa) = 3 x 0.206 = 0.618 kPa.*<br>In the Relative Threshold and Slope Threshold algorithms, pressure is considered to be rising when the current pressure value exceeds the previous pressure value by more than 0.618 kPa. |
| PCRELT | Pressure Change Relative Threshold | 0x0032 | Relative Threshold value configured to 0x32 = 50 counts. The threshold value in kPa is equal to:<br>*Relative Threshold (kPa) = 50 x 0.206 = 10.3 kPa.* |

**Table 1. NBP default configuration after POR*...continued***

| Register | Name | Reset value | Description |
|---|---|---|---|
| PCSLOPET | Pressure Change Slope Threshold | 0x0040 | Slope Threshold value configured to 0x40 = 64 counts. With a sampling period of 135 ms, the slope threshold value in kPa/s is equal to: *Slope Threshold (kPa/s) = 64 x (0.206/128) x [(1000 + 135)/135] = 0.866 kPa/s.* |

The Host can update the NBP register configuration via SPI. The list and addresses of user registers accessible via SPI, also called read/write targets, are provided in the NBP data sheets[2],[3]. The registers indicated as Read Only must not be written by the Host.

After the Host has updated the NBP register configuration, the new configuration is maintained as long as the NBP is supplied. If the NBP is turned off, the registers are configured with their reset value again after POR.

# 3 Configuring the thresholds in the Pressure Change algorithms

## 3.1 General information

Three pressure change algorithms are supported by the NBP. The user can enable any one, two, or three algorithms. When an algorithm is enabled, it is executed at every sampling period after a new valid pressure measurement has been taken and the pressure FIFO updated. When more than one algorithm is enabled at the same time, the enabled algorithms run independently of each other at every sampling period.

Each algorithm has its own flag indicating that the conditions for pressure change have been met:

- STATUS_PCFTF for the Fixed Threshold algorithm,
- STATUS_PCRTF for the Relative Threshold algorithm,
- STATUS_PCSTF for the Slope Threshold algorithm.

When at least one of the flags is set, STATUS_INTF flag is set as well, which triggers a notification to the Host. When a flag is set, it remains set until the Host sets CMD_ACKINTF to clear it.

After STATUS_PCFTF, STATUS_PCRTF or STATUS_PCSTF is set, the pressure FIFO continues to be updated at every sampling period, and the algorithms continue to run. The NBP continues monitoring pressure and detecting new pressure events even after one of the algorithms has converged. This implementation ensures that new pressure events are not missed.

***Important:***

*If the pressure inside the battery pack exceeds the maximum pressure that can be measured by the NBP, the pressure value is set to $P_{max}$ value and the pressure overflow bit POVER in SENSTATUS register is set. Since the NBP is not able to follow the pressure variations above $P_{max}$ value, the pressure change algorithms will not converge. In that case, the Host can rely on SENSTATUS_POVER bit being set to recognize that a pressure event has occurred. When SENSTATUS_POVER bit is set, the NBP triggers a notification to the Host if INTTRIG_SENSERR bit is set.*

*Refer to the data sheets[2],[3] for information on $P_{max}$ value and the pressure ranges supported by the NBP.*

## 3.2 Fixed Threshold algorithm

Setting PCCFG_FTEN bit enables the Fixed Threshold algorithm. Two user-configurable thresholds are used by the algorithm:

- PCDBT, the Debounce Threshold. The Debounce Threshold value is common to the three algorithms. STATUS_PCFTF flag is set when the Debounce counter of the Fixed Threshold algorithm exceeds the Debounce Threshold.
- PCFIXT, the Fixed Threshold. The Debounce counter of the Fixed Threshold algorithm is incremented when the pressure value exceeds the Fixed Threshold value.
  The threshold value in PCFIXT register is expressed in counts. The value can be converted to kPa using the pressure transfer function:
  *Fixed Threshold in kPa = ($\Delta P_{MAX-MIN}$ × PCFIXT) + 39.6.*
  The parameter $\Delta P_{MAX-MIN}$ is specified in the data sheets.[2],[3]

Example:

**Requirements:**

- The Fixed Threshold flag shall be set when pressure exceeds150 kPa during at least four sampling periods (that is, four sampling periods or more).

**Configuration:**

- From the data sheet, $\Delta P_{MAX-MIN}$ = 0.206 kPa/LSB.
- PCFIXT value in counts is calculated following the formula:
  PCFIXT = (150 – 39.6) / 0.206 = 536 counts.
- PCDEBT is set to value (4 – 1) = 3.

**Resulting flow:**

- When the pressure value exceeds 536 counts, the Debounce counter is incremented. When the pressure value is equal to or below 536 counts, the Debounce counter is decremented.
- When the Debounce counter is strictly greater than value 3, that is, when it reaches value 4, STATUS_PCFTF flag is set.

***Note:*** *If PCDEBT is set to 0, STATUS_PCFTF flag is set as soon as pressure exceeds the threshold. If PCDEBT is set to a value strictly greater than 0, the pressure value needs to exceed the threshold during more than one sampling period for STATUS_PCFTF flag to be set. Configuring a PCDEBT value greater than 0 prevents STATUS_PCFTF from being set when a single measurement exceeds the threshold due to a coincidental event, such as noise, affecting the measurement.*

## 3.3 Relative Threshold algorithm

Setting PCCFG_RTEN bit enables the Relative Threshold algorithm. Three user-configurable thresholds are used by the algorithm:

- PCDBT, the Debounce Threshold. The Debounce Threshold value is common to the three algorithms. STATUS_PCRTF flag is set when the Debounce counter of the Relative Threshold algorithm exceeds the Debounce Threshold.
- PCMINT, the Minimum Threshold. The Minimum Threshold value is common to the Relative Threshold and Slope Threshold algorithms. In these algorithms, pressure is considered to be increasing only if the current pressure value exceeds the previous pressure value by PCMINT counts.
  PCMINT is expressed in pressure counts. The value can be converted to kPa using the formula:
  *Minimum Threshold in kPa = $\Delta P_{MAX-MIN}$ × PCMINT.*
  The parameter $\Delta P_{MAX-MIN}$ is specified in the data sheets[2],[3].
- PCRELT, the Relative Threshold. The Debounce counter of the Relative Threshold algorithm is incremented when the pressure increase exceeds the Relative Threshold value.
  The threshold value in PCRELT register is expressed in counts. The value can be converted to kPa using the formula:
  *Relative Threshold in kPa = $\Delta P_{MAX-MIN}$ × PCRELT.*
  The parameter $\Delta P_{MAX-MIN}$ is specified in the data sheets[2],[3].

Example:

**Requirements:**

- The Relative Threshold flag shall be set when pressure has increased by more than 50 kPa during at least five sampling periods (that is, five sampling periods or more).
- Pressure shall be considered to be increasing from one sampling period to the next only if the pressure increase is greater than 0.5 kPa. A pressure increase equal to or below 0.5 kPa between two consecutive samples shall be considered to be sensor drift and not an actual pressure increase.

**Configuration:**

- From the data sheet, $\Delta P_{MAX-MIN}$ = 0.206 kPa/LSB.
- PCRELT value is calculated following the formula:
  PCRELT = 50 / 0.206 = 242 counts.
- PCDEBT is set to value (5 – 1) = 4.
- PCMINT value is calculated following the formula:
  PCMINT = 0.5 / 0.206 = 2 counts.

**Resulting flow:**

- Pressure is considered to be increasing when the current pressure value exceeds the previous pressure value by three counts or more.
- When pressure has been increasing, the pressure increase is calculated as the difference between the current pressure and the reference pressure. If the pressure increase is greater than 242 counts, the Debounce counter is incremented. The Debounce counter is decremented otherwise.
- When the Debounce counter is strictly greater than value 4, that is, when it reaches value 5, STATUS_PCRTF flag is set.

***Note:*** *If PCDEBT is set to 0, STATUS_PCRTF flag is set as soon as the pressure increase exceeds the threshold. If PCDEBT is set to a value strictly greater than 0, the pressure increase needs to exceed the threshold during more than one sampling period for STATUS_PCRTF flag to be set. Configuring a PCDEBT value greater than 0 prevents STATUS_PCRTF from being set when a single measurement causes the pressure increase to exceed the threshold due to a coincidental event, such as noise, affecting the measurement.*

## 3.4 Slope Threshold algorithm

Setting PCCFG_STEN bit enables the Slope Threshold algorithm. Three user-configurable thresholds are used by the algorithm:

- PCDBT, the Debounce Threshold. The Debounce Threshold value is common to the three algorithms. When the Debounce counter of the Slope Threshold algorithm exceeds the Debounce Threshold, the pressure slope is calculated and compared to the Slope Threshold.
- PCMINT, the Minimum Threshold. The Minimum Threshold value is common to the Relative Threshold and Slope Threshold algorithms. In these algorithms, pressure is considered to be increasing only if the current pressure value exceeds the previous pressure value by PCMINT counts.
  PCMINT is expressed in counts. The value can be converted to kPa using the formula:
  *Minimum Threshold in kPa* = $\Delta P_{MAX-MIN}$ × *PCMINT.*
  The parameter $\Delta P_{MAX-MIN}$ is specified in the data sheets[2],[3].
- PCSLOPET, the Slope Threshold. STATUS_PCSTF flag is set when the slope value exceeds the Slope Threshold.
  The slope is calculated using the formula:

$$Slope = \frac{\Delta P \times 128}{number\ of\ sampling\ periods + 1}$$

The threshold value in PCSLOPET register is expressed in pressure counts per number of sampling periods. The value can be converted to kPa/s using the formula:

$$Slope\ \ Threshold\ \ in\ \ \ kPa/s = \text{PCSLOPET} \times \frac{\Delta P_{\text{MAX-MIN}}}{128} \times \frac{1000+sampling\_period\_ms}{sampling\_period\_ms}$$

The parameter *sampling_period_ms* is the sampling period value expressed in milliseconds. The parameter $\Delta P_{MAX-MIN}$ is specified in the data sheets[2],[3].

Example:

**Requirements:**

- A new pressure measurement shall be taken every 40 ms for the NBP8 or every 50 ms for the NBP9.
- The Slope Threshold flag shall be raised when pressure has increased with a rate equal or greater than 15 kPa/s.
- Pressure shall be considered to be increasing from one sampling period to the next only if the pressure increase is greater than 0.5 kPa. A pressure increase below 0.5 kPa between two consecutive samples shall be considered to be sensor drift and not an actual pressure increase.
- The slope shall be calculated when pressure has been increasing for at least 10 sampling periods.

**Configuration:**

- From the data sheet, $\Delta P_{MAX-MIN}$ = 0.206 kPa/LSB.
- In the NBP8, PSP register is set to value 2. In the NBP9, MODECFG_MODE bit is set to 1.
- In the NBP8, a pressure increase of 15 kPa over 1 second corresponds to an increase of (15 / 0.206) = 73 counts over (1 / 0.04) = 25 sampling periods. PCSLOPET value is calculated following the formula: PCSLOPET = 73 x 128 / (25 + 1) = 359 counts.
  In the NBP9, a pressure increase of 15 kPa over 1 second corresponds to an increase of (15 / 0.206) = 73 counts over (1 / 0.05) = 20 sampling periods. PCSLOPET value is calculated following the formula: PCSLOPET = 73 x 128 / (20 + 1) = 445 counts.
- PCDEBT is set to value (10 – 1) = 9.
- PCMINT value is calculated following the formula: PCMINT = 0.5 / 0.206 = 2 counts.

**Resulting flow:**

- Pressure is considered to be increasing when the current pressure value exceeds the previous pressure value by three counts or more.
- When pressure has been increasing, the Debounce counter is incremented. The Debounce counter is decremented otherwise.
- When the Debounce counter is strictly greater than value 9, that is, when it reaches value 10, the slope is calculated following the formula
  *Slope = ΔP * 128 / (number of sampling periods + 1).*
  STATUS_PCSTF is set if the slope value is equal or greater than 359 counts for the NBP8 and 445 counts for the NBP9.

# 4 STATUS and SENSTATUS registers

## 4.1 Purpose

STATUS register is a read-only register that gives information on the status of the latest measurements, latest self-test, and latest firmware verification, and indicates whether a pressure change algorithm has converged. When the NBP notifies the Host of an event, the Host should read STATUS register to know the origin of the notification. If STATUS_SENSF is set, indicating that an error or warning occurred during sensor measurements, SENSTATUS read-only register should be read for additional information. In addition to reading STATUS and SENSTATUS registers, the Host should set CMD_ACKINTF to clear the registers.

## 4.2 Bit fields update

The bit fields of STATUS and SENSTATUS registers are updated following different strategies:

- STATUS_PCFTF is set when the Fixed Threshold algorithm has converged. The bit can only be cleared by the Host setting CMD_ACKINTF bit.
  When PCFTF is set, STATUS_INTF is set as well.
- STATUS_PCRTF is set when the Relative Threshold algorithm has converged. The bit can only be cleared by the Host setting CMD_ACKINTF bit.
  When PCRTF is set, STATUS_INTF is set as well.
- STATUS_PCSTF is set when the Slope Threshold algorithm has converged. The bit can only be cleared by the Host setting CMD_ACKINTF bit.
  When PCSTF is set, STATUS_INTF is set as well.
- STATUS_FVF is set when the latest firmware verification completed with errors. If INTTRIG_FVERR is set, STATUS_INTF is set as well.
  STATUS_FVF is cleared when the latest firmware verification completed with no error, or by the Host setting CMD_ACKINTF.
  Figure 1 shows how the flags are updated during the execution of firmware verification.

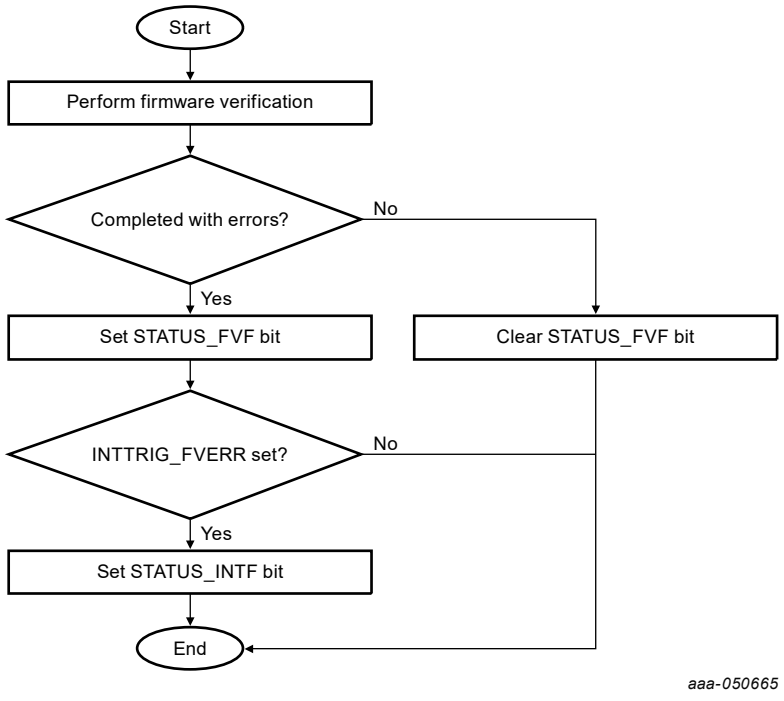


**Figure 1. Bit fields update during firmware verification**

- STATUS_PSTF is set when the latest Pcell self-test completed with errors. If INTTRIG_STERR is set, STATUS_INTF is set as well.
  STATUS_PSTF is cleared when the latest Pcell self-test completed with no error, or by the Host setting CMD_ACKINTF.
  Figure 2 shows how the flags are updated during the execution of Pcell self-test.

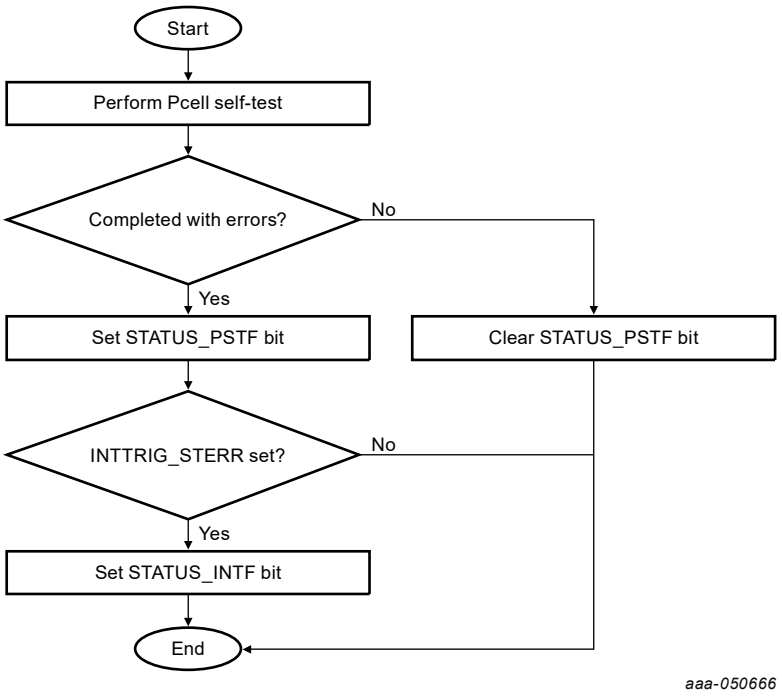


**Figure 2. Bit fields update during Pcell self-test**

- STATUS_ADCSTF is set when the latest ADC self-test completed with errors. If INTTRIG_STERR is set, STATUS_INTF is set as well.
  STATUS_ADCSTF is cleared when the latest ADC self-test completed with no error, or by the Host setting CMD_ACKINTF.
  Figure 3 shows how the flags are updated during the execution of ADC self-test.

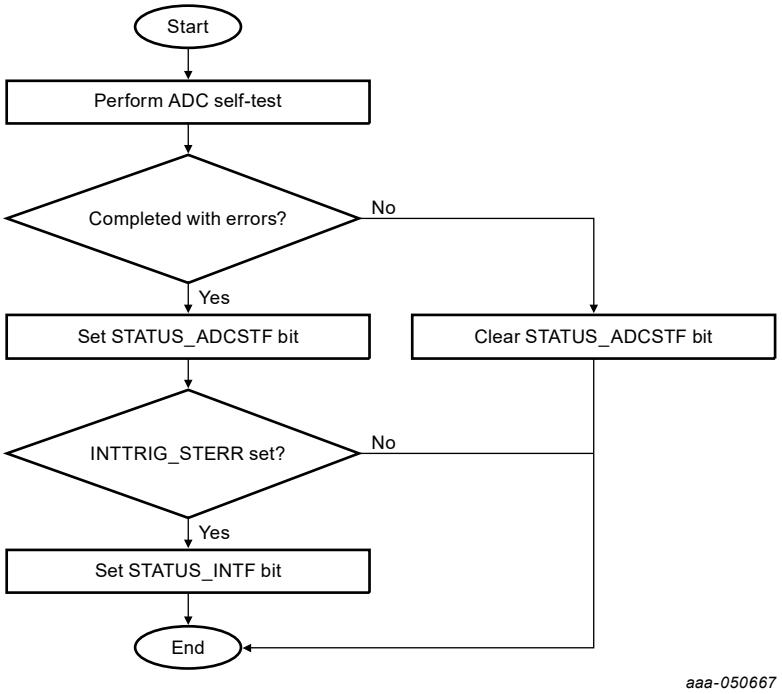


**Figure 3. Bit fields update during ADC self-test**

- STATUS_SENSF and SENSTATUS bits are set when the latest sensor measurements completed with errors or warnings. These bits are updated at every sampling period after the measurements have been performed.

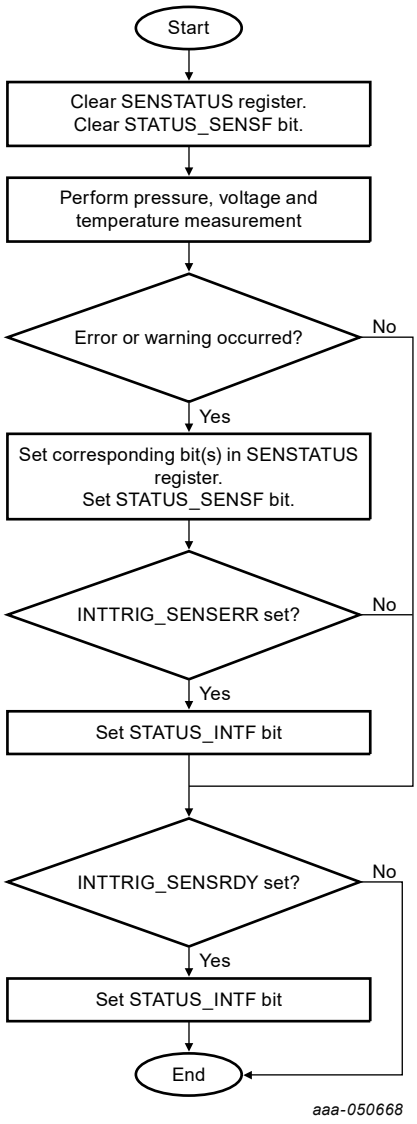


**Figure 4. Bit fields update during sensor measurements**

- STATUS_INTF can only be cleared by the Host setting CMD_ACKINTF bit.

Figure 5 provides a summary of the update strategies for STATUS and SENSTATUS bit fields.

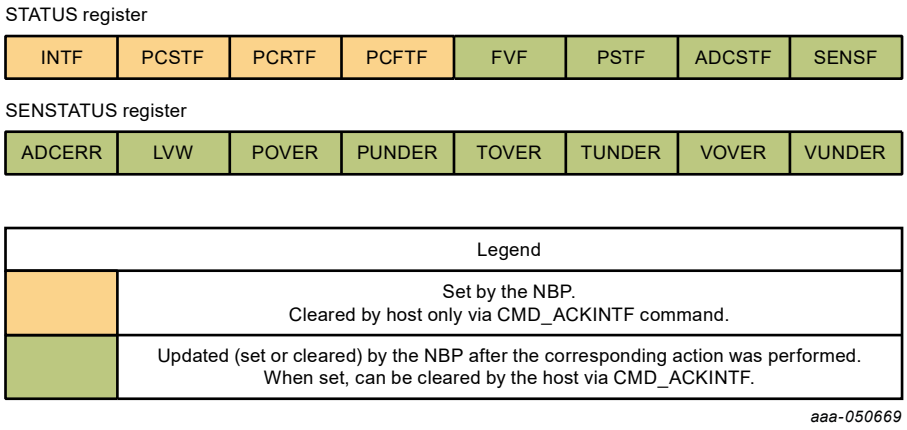


**Figure 5. Update strategies for STATUS and SENSTATUS bit fields**

## 4.3 Note on STATUS_INTF flag

When STATUS_INTF is set, the NBP sends a notification to the Host. STATUS_INTF bit can only be cleared by the Host setting CMD_ACKINTF bit. If the Host does not set CMD_ACKINTF bit to clear the flags, then STATUS_INTF remains set and the NBP continues sending periodic notifications to the Host. It is therefore important that the Host sets CMD_ACKINTF when the NBP sends a notification.

When the Host reads STATUS register and that STATUS_INTF is set, the Host can generally know the source of the notification via the other bit fields of the register. However, it is possible that only STATUS_INTF flag bit is set, and the other bit fields are all clear. Such situation can occur in the following two cases:

- If INTTRIG_SENSRDY is set, then STATUS_INTF bit is set at every sampling period after the sensor measurements have been taken. In this case, only INTF bit of STATUS register is set. If no warning or error occurred during sensor measurements, the other bit fields are all clear.
- The following flow occurred:
  1. FVF, PSTF, ADCSTF, or SENSF bit of STATUS register was set and caused INTF bit to be set as well, which triggered a notification to the Host.
  2. The Host did not set CMD_ACKINTF following the notification.
  3. FVF, PSTF, ADCSTF, or SENSF bit was later cleared by the NBP application.

  In that case, only INTF bit remains set. The Host therefore knows that an error or warning has occurred but was later resolved.

## 4.4 Handling errors

If STATUS_FVF, STATUS_PSTF or STATUS_ADCSTF is set, perform at least one retry to see if the error is persistent. To trigger firmware verification, set CMD_FV. To trigger Pcell self-test, set CMD_PST. To trigger ADC self-test, set CMD_ADCST.

STATUS_SENSF bit set indicates that an error or warning occurred during the latest sensor measurements. The cause of the error or warning is indicated in SENSTATUS register:

- SENSTATUS_ADCERR indicates that either pressure, temperature, or voltage measurement could not be taken because the ADC failed to converge. When such error occurs, the Host should discard the sensor measurement values and trigger an ADC self-test. Following an ADC error, other flags in SENSTATUS register may be set in addition to ADCERR. The Host should discard the other flags and consider they are due to the ADC error.
- SENSTATUS_LVW indicates that the supply voltage dropped below the recommended range during pressure, temperature, or voltage measurement. The accuracy of the measurements is therefore not guaranteed.

- SENSTATUS_POVER and PUNDER indicate that the pressure value is out-of-range and could not be measured.
- SENSTATUS_TOVER and TUNDER indicate that the temperature value is out-of-range and could not be measured.
- SENSTATUS_VOVER and VUNDER indicate that the voltage value is out-of-range and could not be measured.

***Note:*** *The final pressure value stored in the pressure FIFO is the result of a compensation routine that takes as input the raw pressure, temperature, and voltage. The purpose of the compensation is to correct the known pressure cell deviation to temperature and voltage.*

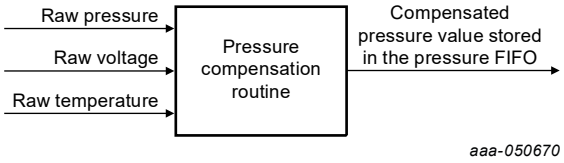


**Figure 6. Input and output to the pressure compensation routine**

Since the pressure compensation routine takes as input the voltage and temperature values, users should understand that invalid voltage, or temperature values lead to an invalid compensated pressure value. As a consequence, when any flag of the SENSTATUS register is set, the pressure value is considered to be invalid.

# 5 Handling an NBP transfer request

## 5.1 Handling a transfer request from the NBP8

When STATUS_INTF flag is set, the NBP8 triggers a notification to the Host. The notification sequence depends on whether the PS ENABLE feature is enabled or not. Configuring PINCFG register to value 0x04 or 0x05 enables the PS ENABLE feature. All other values disable the feature.

When the PS ENABLE feature is disabled, the NBP8 performs the following sequence:

1. Enable SPI.
2. Generation of a pulse on the INT pin.
3. Halt CPU.
4. Wait for CPU to restart. The CPU is restarted and SPI disabled when the Host clears SPIOPS register or when the 2048 ms timeout occurs.

Figure 7 shows the pulse generated on the INT/READY pin, followed by the SPI transfers.

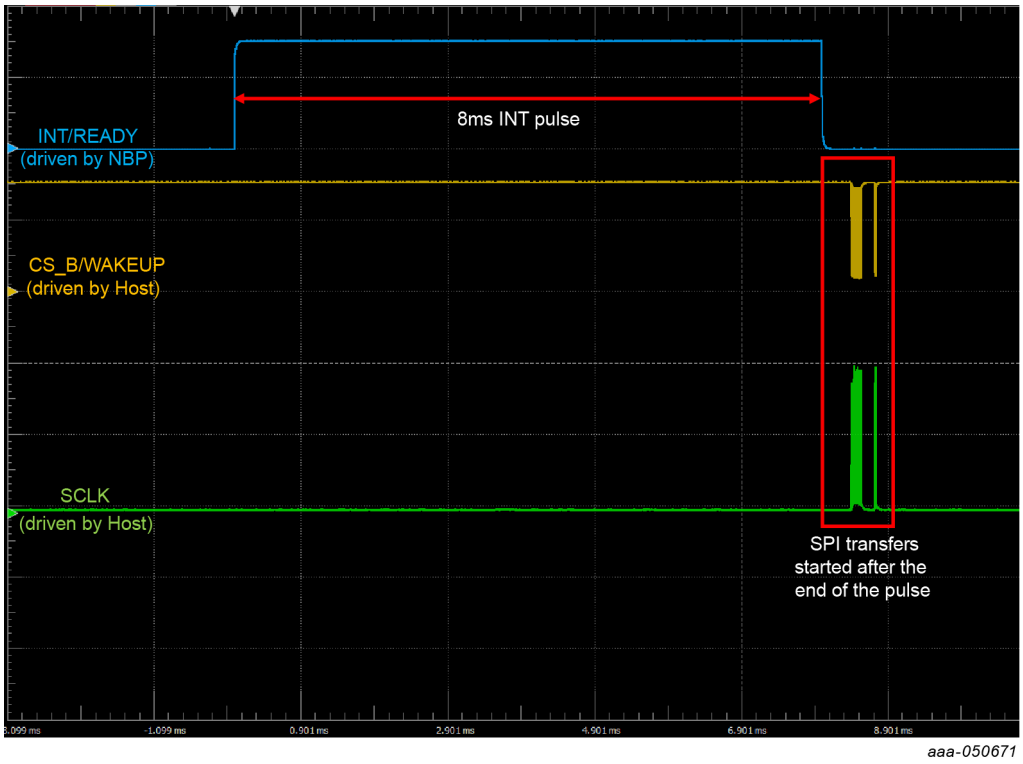


**Figure 7. Signals during an NBP8 transfer request**

When the PS ENABLE feature is enabled, the NBP8 performs the following sequence:

1. Assertion of PS ENABLE pin.
2. Delay of 200 ms.
3. Enable SPI.
4. Generation of a pulse on the INT pin.
5. Halt CPU.
6. Wait for CPU to restart. The CPU is restarted and SPI disabled when the Host clears SPIOPS register or when the 2048 ms timeout occurs.
7. Drive PS ENABLE pin to idle state.

Figure 8 shows the assertion of PS ENABLE pin followed by a 200 ms delay. The pulse is then performed on the INT/READY pin, followed by SPI transfers.

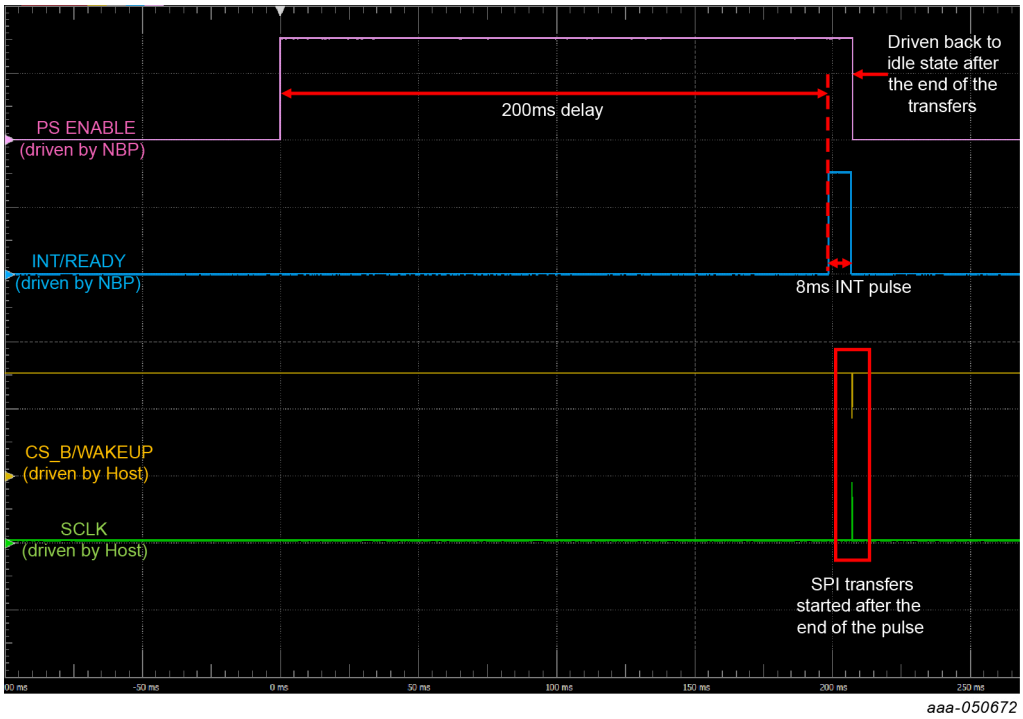


**Figure 8. Signals during an NBP8 transfer request with PINCFG = 0x05**

In both cases, the NBP8 halts its CPU after the completion of the INT pulse. The Host application must wait for the pulse to end before starting SPI communication. If SPI communication is started while the NBP CPU is not halted, memory contention issues occur and the SPI commands are not executed by the NBP.

***Note:*** *When the NBP8 CPU is halted, the application is not running anymore. In order to ensure that the NBP8 application resumes as soon as possible after the end of the SPI transfers, the Host must always clear SPIOPS register at the end of the transfer sequence.*

## 5.2 Handling a transfer request from the NBP9

When STATUS_INTF flag is set, the NBP9 triggers a notification to the Host. The notification sequence depends on whether the PS ENABLE feature is enabled or not. Configuring PINCFG register to value 0x04 or 0x05 enables the PS ENABLE feature. All other values disable the feature.

When the PS ENABLE feature is disabled, the NBP9 performs the following sequence:

1. Generation of a pulse on the INT pin.
2. Generation of the PWM signal.
3. Wait for the Host to lower CS_B/WAKEUP pin.
4. Stop PWM generation.
5. Enable SPI.
6. Assert INT/READY pin.
7. Halt CPU.
8. Wait for CPU to restart. The CPU is restarted and SPI disabled when the Host clears SPIOPS register or when the 2048 ms timeout occurs.
9. Drive INT/READY pin back to idle state.

Figure 9 shows the pulse generated on the INT/READY pin, followed by the PWM generation until the Host lowers CS_B/WAKEUP pin. SPI transfers are performed after the NBP asserts the INT/READY pin.

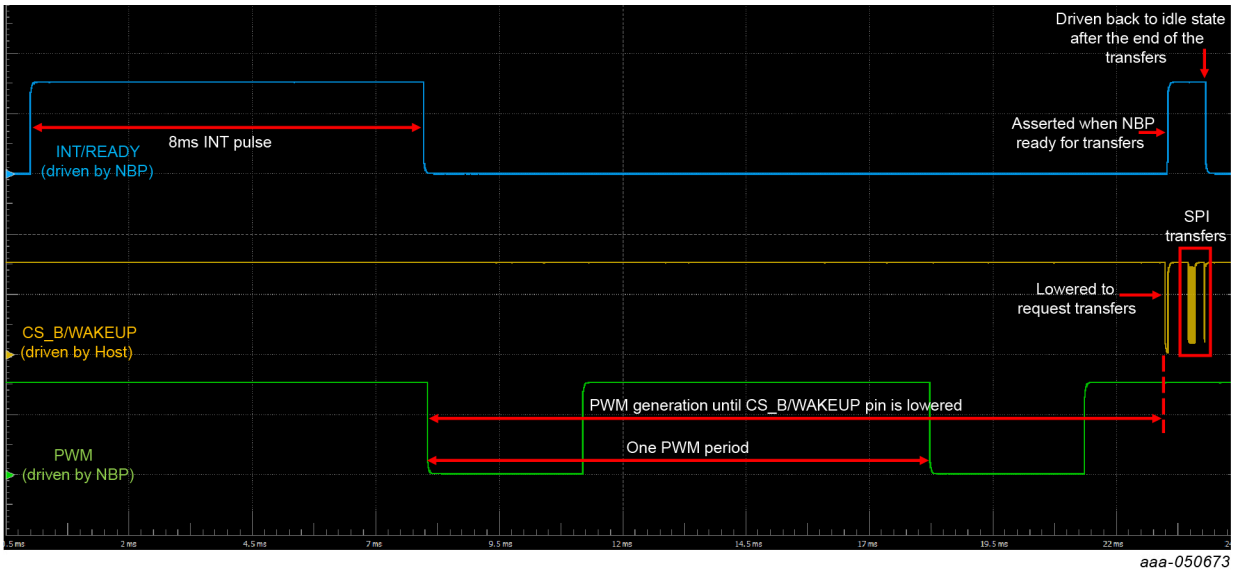


**Figure 9. Signals during an NBP9 transfer request**

When the PS ENABLE feature is enabled, the NBP9 performs the following sequence:

1. Assertion of PS ENABLE pin.
2. Delay of 200 ms.
3. Generation of a pulse on the INT pin.
4. Generation of the PWM signal.
5. Wait for the Host to lower CS_B/WAKEUP pin.
6. Stop PWM generation.
7. Enable SPI.
8. Assert INT/READY pin.
9. Halt CPU.
10. Wait for CPU to restart. The CPU is restarted and SPI disabled when the Host clears SPIOPS register or when the 2048 ms timeout occurs.
11. Drive INT/READY and PS ENABLE pins back to idle state.

Figure 10 shows the assertion of PS ENABLE pin followed by a 200 ms delay. A pulse is then generated on the INT/READY pin, followed by the PWM generation until the Host lowers CS_B/WAKEUP pin. SPI transfers are performed after the NBP asserts the INT/READY pin.

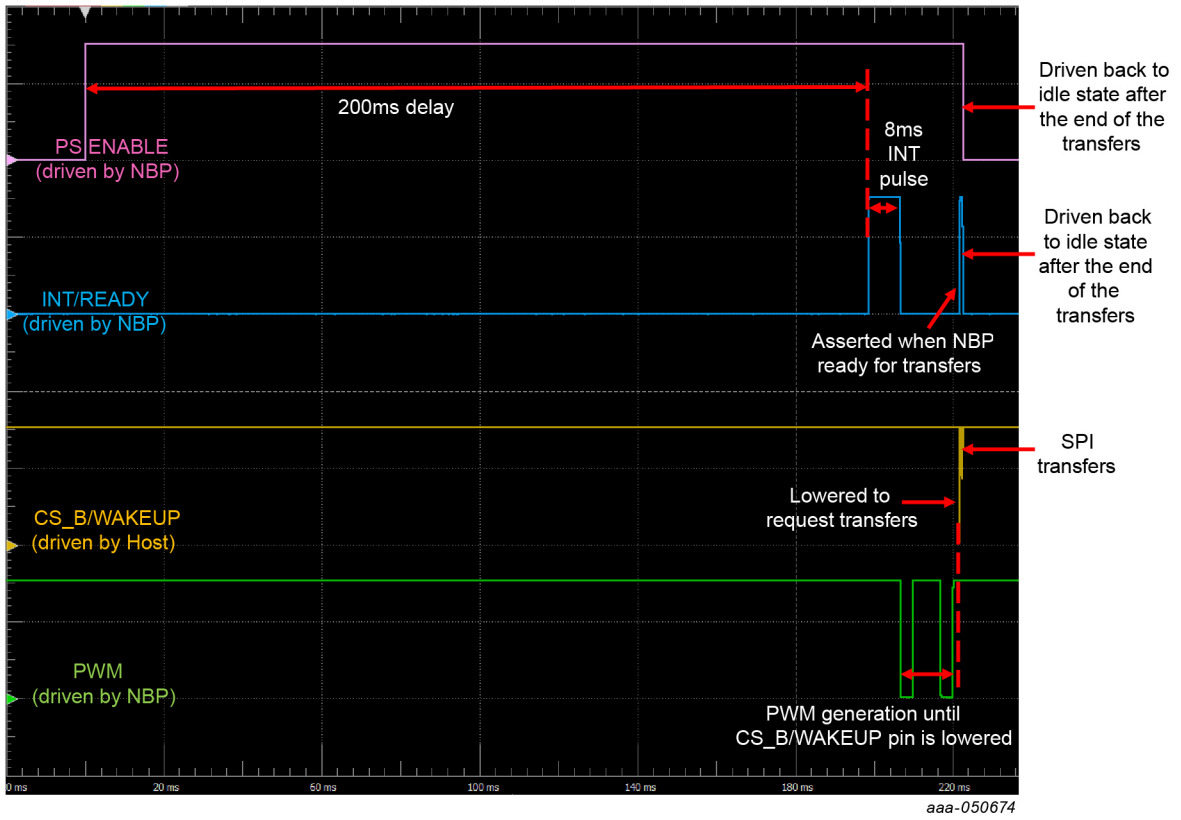


**Figure 10. Signals during an NBP9 transfer request with PINCFG = 0x05**

In both cases, the NBP9 continues generating the PWM signal as long as the Host maintains the CS_B/WAKEUP pin at high state. If the Host does not lower the CS_B/WAKEUP pin within the 2048 ms timeout time, the timeout occurs, the PWM generation is stopped, and the NBP9 application continues. In that case, SPI is not enabled.

After the Host lowers the CS_B/WAKEUP pin, the NBP9 stops the PWM generation, enables SPI, asserts the INT/READY pin, and halts its CPU. The Host application must wait for the INT/READY pin to be asserted before starting SPI communication. SPI is disabled after the CPU is restarted.

***Note:*** *When the NBP9 CPU is halted, the application is not running anymore. In order to ensure that the NBP9 application resumes as soon as possible after the end of the SPI transfers, the Host must always clear SPIOPS register at the end of the transfer sequence.*

***Note:*** *When the Host is answering a transfer request from the NBP9, it should be noted that the first SPI command sent by the Host is ignored by the NBP9. The first command transmitted should therefore be a dummy READ command. The NBP9 starts executing the SPI commands from the second command onwards.*

# 6 Triggering a transfer request with the NBP8 and NBP9

## 6.1 Preliminary check

Before triggering a transfer request, the Host must first check that the NBP has not already started sending a notification. For that, the Host must check that the INT/READY pin, and PS ENABLE pin, if enabled, are at idle state. If INT/READY pin and/or PS ENABLE pin is asserted, this indicates that the NBP has started the notification sequence. In that case, the Host should follow the sequence described in previous section to answer the notification.

## 6.2 Flow

To trigger SPI communication with the NBP, the host needs to lower the CS_B/WAKEUP pin and then wait for the NBP to assert the INT/READY pin as acknowledge. On the NBP side, SPI is enabled only when the INT/READY pin is asserted. The Host must wait for the INT/READY pin to be asserted before starting the SPI transfers.

The duration between the moment the Host lowers the CS_B/WAKEUP pin and the moment the NBP asserts the INT/READY pin is variable. It depends on the action that the NBP is performing when the CS_B/WAKEUP pin is lowered:

- If the NBP is in sleep: the NBP needs to wake up first. After wake-up, some initialization instructions are performed before the NBP enables SPI, asserts the INT/READY pin, and halts its CPU. The duration of these actions is around 120 µs.
- If the NBP is performing periodic events: the NBP first completes all periodic events (self-test, sensor measurements, and pressure change algorithms) before enabling SPI, asserting the INT/READY pin, and halting its CPU. The duration before the INT/READY pin is asserted depends on the remaining actions to complete. In the worst-case scenario, the transfer request occurs when the NBP has just started the self-test. In this scenario, the NBP needs to complete all events before answering the SPI request, which takes around 8 ms.
- If the NBP is performing actions that have been triggered via the CMD register: the NBP first completes all actions before enabling SPI, asserting the INT/READY pin, and halting its CPU. The duration before the INT/READY pin is asserted depends on the actions requested by the Host. The longest action is firmware verification, with an execution time around 132 ms.

After the NBP has halted its CPU, the CPU remains halted until the Host clears SPIOPS register or until the 2048 ms timeout occurs. SPI is disabled and the INT/READY pin driven back to idle state after the CPU has restarted.

***Note:*** *When the NBP CPU is halted, the application is not running anymore. In order to ensure that the NBP application resumes as soon as possible after the end of the SPI transfers, the Host must always clear SPIOPS register at the end of the transfer sequence.*

Figure 11 shows the Host lowering the CS_B/WAKEUP pin, and then waiting for the assertion of the INT/READY pin, which takes 120 µs. The INT/READY pin is driven back to idle state by the NBP after the end of the transfers.

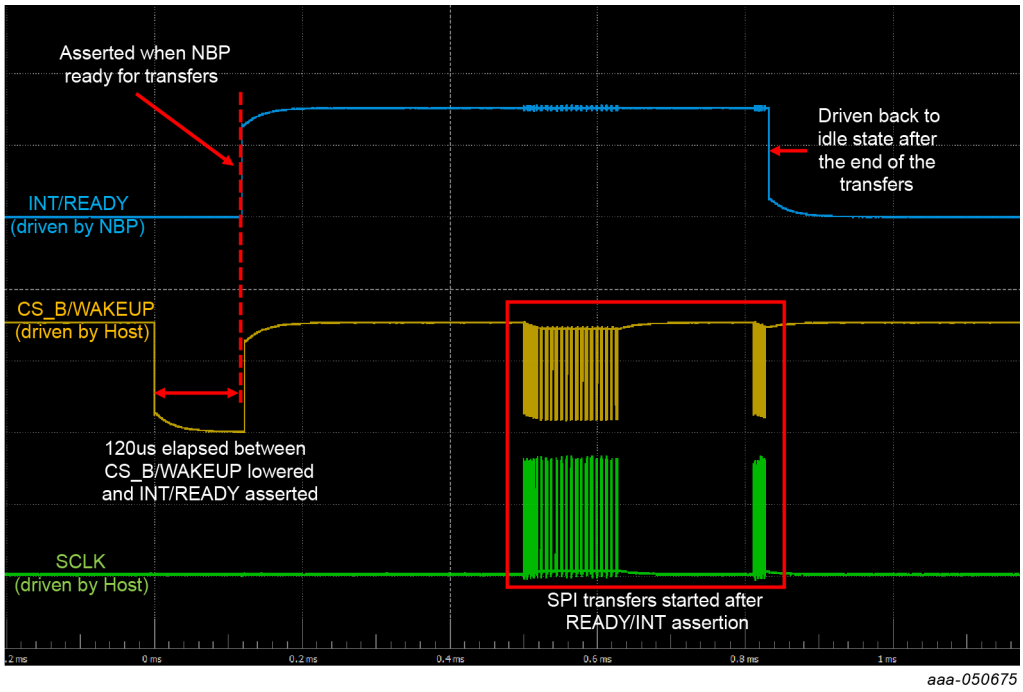


**Figure 11. Signals during transfers requested by the host**

## 6.3 Points of attention

When the Host is triggering a transfer request, there are two points of attention:

- The first SPI command sent by the Host is ignored by the NBP. The first command transmitted should be a dummy READ command. The NBP starts executing the SPI commands from the second command onwards.
- By coincidence, it can happen that the Host and the NBP request transfers at the same time. When this happens, the NBP asserts the INT/READY pin at the same time the Host lowers the CS_B/WAKEUP pin. After lowering the CS_B/WAKEUP pin, the Host sees that the INT/READY pin is asserted and incorrectly assumes that the NBP is acknowledging the transfer request whereas it is generating the pulse. As a result, the Host may start the SPI transfers while the NBP is generating the pulse and has not yet halted its CPU. In case of the NBP9, SPI is not enabled yet. The consequence is that memory contention issues occur in case of the NBP8, and SPI commands are discarded in case of the NBP9. In both cases, the commands transferred by the Host are not executed.
  In order to detect if the NBP has already enabled SPI and halted its CPU, a good practice for the Host is to read SPIOPS register. If BIT2 of SPIOPS register is set, this indicates that the NBP CPU is halted, so the NBP is fully ready for SPI communication. If BIT2 of SPIOPS register is clear, this indicates that the NBP CPU is still running. In this case, the Host should wait for the INT pulse to end and handle the NBP transfer request.

# 7 SPI transfers

During SPI communication, the Host can send READ and WRITE commands to the NBP. The NBP replies to the command during the next transfer. The NBP response contains five bits indicating the status of the transfer or the command execution. The Host should always read the status bits to know when an error occurred. To decode the status, refer to the data sheet[2],[3]. An excel file allowing to encode and decode SPI commands and responses is provided as associated file to this application note.

When an error occurs during a transfer, the NBP discards the next transfer in order to clear the error. When a transfer is discarded by the NBP, the READ or WRITE command is ignored.

Table 2 contains examples of SPI commands and responses in case of transfers following a notification from the NBP8 due to STATUS_PCFTF flag set.

**Table 2. Examples of SPI commands and responses following a notification from the NBP8**

| Transfer number | SPI command from the Host to the NBP8 | | SPI response from the NBP8 to the Host | |
|---|---|---|---|---|
| | Command | Description | Response | Description |
| 0 | 0x0154 | Read address 0x55<br>(STATUS) | 0x2002 | First command after reset (sleep exit is considered a reset source) |
| 1 | 0x0158 | Read address 0x56<br>(SENSTATUS) | 0x0243 | Status = 0b00000, Data = 0x90<br>(STATUS_INTF and PCFTF set) |
| 2 | 0x815F | Write at address 0x57<br>(CMD) | 0x0000 | Status = 0b00000, Data = 0x00<br>(SENSTATUS clear) |
| 3 | 0x8200 | Write value 0x80<br>(set CMD_ACKINTF) | 0x815F | Address = 0x57<br>(CMD) |
| 4 | 0x80E3 | Write at address 0x38<br>(SPIOPS) | 0x8200 | Status = 0b00000, Data = 0x80<br>(CMD_ACKINTF set) |
| 5 | 0x8002 | Write value 0x00<br>(clear CORE_TR_HOLD) | 0x80E3 | Address = 0x38<br>(SPIOPS) |

***Note:*** *The NBP CPU restarts as soon as the NBP receives the command in transfer number 5. When the NBP CPU is restarted, SPI is disabled. The Host should not perform additional transfers after transfer number 5 because the NBP SPI block is disabled.*

Table 3 contains examples of SPI commands and responses in case of transfers requested by the Host.

**Table 3. Examples of SPI commands and responses following transfers requested by the Host**

| Transfer number | SPI command from the Host to the NBP | | SPI response from the NBP to the Host | |
|---|---|---|---|---|
| | Command | Description | Response | Description |
| 0 | 0x0140 | Dummy command | 0x1002 | Status = 0b00100<br>(clock fault error) |
| 1 | 0x0140 | Read address 0x50<br>(NBP8: PSP<br>NBP9: MODECFG) | 0x2002 | Status = 0b01000<br>(command ignored) |
| 2 | 0x814E | Write at address 0x53<br>(INTTRIG) | NBP8: 0x0011<br>NBP9: 0x0000 | Status = 0b00000<br>NBP8: Data = 0x04<br>NBP9: Data = 0x00 |
| 3 | 0x80FE | Write value 0x3F (set all bits) | 0x814E | Address = 0x53 (INTTRIG) |

Table 3. Examples of SPI commands and responses following transfers requested by the Host...*continued*

| Transfer number | SPI command from the Host to the NBP | | SPI response from the NBP to the Host | |
|---|---|---|---|---|
| | Command | Description | Response | Description |
| 4 | 0x80E3 | Write at address 0x38 (SPIOPS) | 0x80FE | Status = 0b00000, Data = 0x3F (all bits set) |
| 5 | 0x8002 | Write value 0x00 (clear CORE_TR_HOLD) | 0x80E3 | Address = 0x38 (SPIOPS) |

***Note:*** *The NBP CPU restarts as soon as the NBP receives the command in transfer number 5. When the NBP CPU is restarted, SPI is disabled. The Host should not perform additional transfers after transfer number 5 because the NBP SPI block is disabled.*

## 8 Decoding the PWM with the NBP9

The NBP9 generates a PWM signal in two cases:

- When STATUS_INTF is set: after the completion of the INT pulse, the PWM signal is generated continuously until the Host lowers the CS_B/WAKEUP pin or until the 2048 ms timeout occurs.
  Since the PWM signal is generated continuously, the Host can calculate the PWM duty cycle by monitoring the PWM signal between any two falling edges (when MODECFG_PWMPOL = 0) or any two rising edges (when MODECFG_PWMPOL = 1).
  Figure 12 shows the PWM signal generated continuously between the end of the INT pulse and CS_B/WAKEUP pin lowered by the Host.

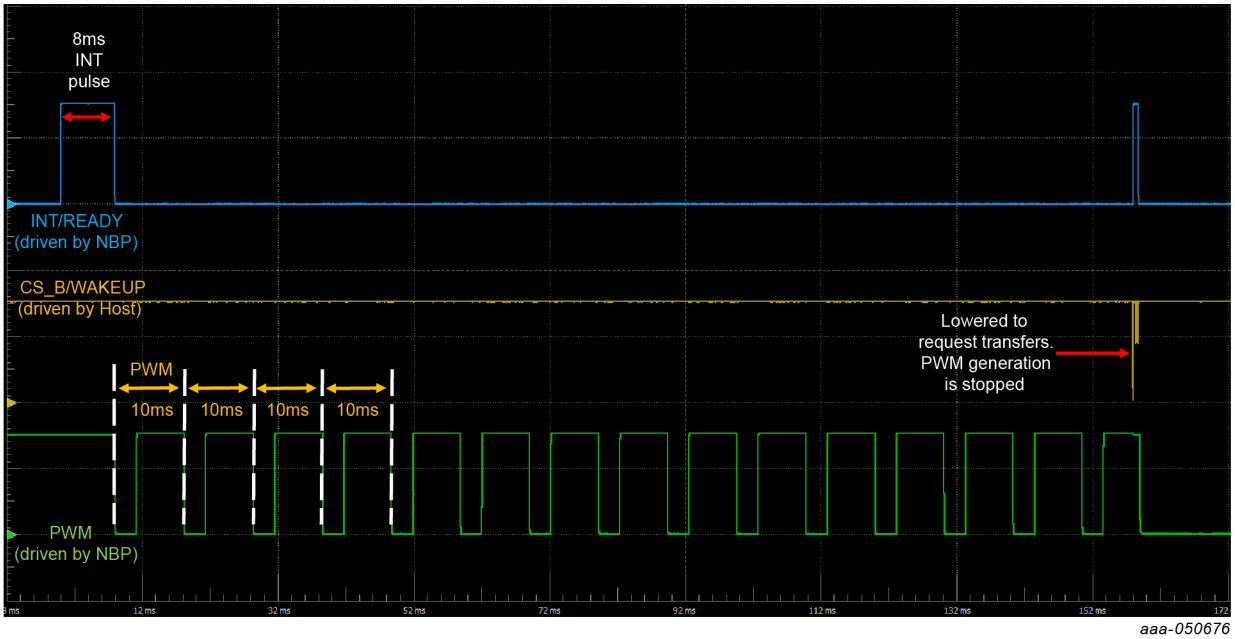


**Figure 12. PWM generation following a notification, with MODECFG_PWMPOL = 0**

- In NORMAL mode, when STATUS_INTF is not set: the PWM signal is generated during four PWM periods after the periodic events (self-test, sensor measurements, and pressure change algorithms) have completed. The PWM signal is not generated while the periodic events are being executed.
  In NORMAL mode, the PWM signal is not generated continuously. As a result, after monitoring the PWM signal between two falling edges (when MODECFG_PWMPOL = 0) or two rising edges (when

MODECFG_PWMPOL = 1), the Host must check that the PWM period is valid before calculating the duty cycle.
A valid PWM period has a duration around 10 ms. Taking into account the NBP internal clock drift, a valid period can last between 9 ms and 12 ms. A duration longer than 12 ms corresponds to a period during which the periodic events are performed in addition to the generation of the PWM signal. As a result, when the Host measures a duration between two falling or two rising edges longer than 12 ms, the duty cycle must not be calculated as the duration does not correspond to a valid PWM period.
Figure 13 shows the PWM signal when the NBP is in Normal mode and self-test is performed at every sampling period.

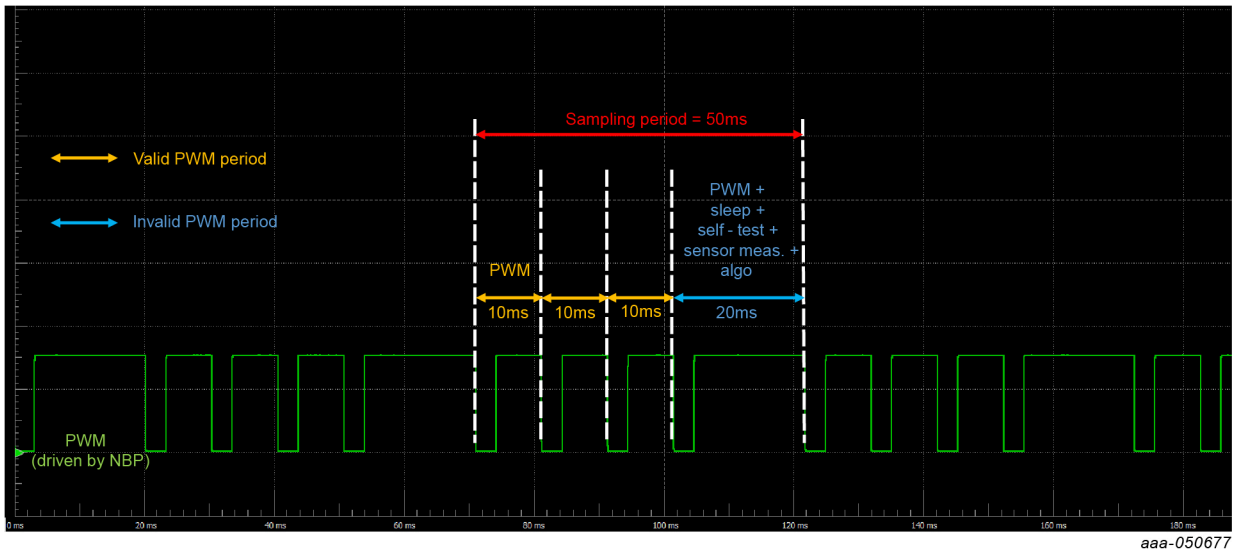


**Figure 13. PWM signal in Normal mode when self-test is performed at every sampling period, with MODECFG_PWMPOL = 0**

Figure 14 shows the PWM signal when the NBP is in Normal mode and self-test is not performed.

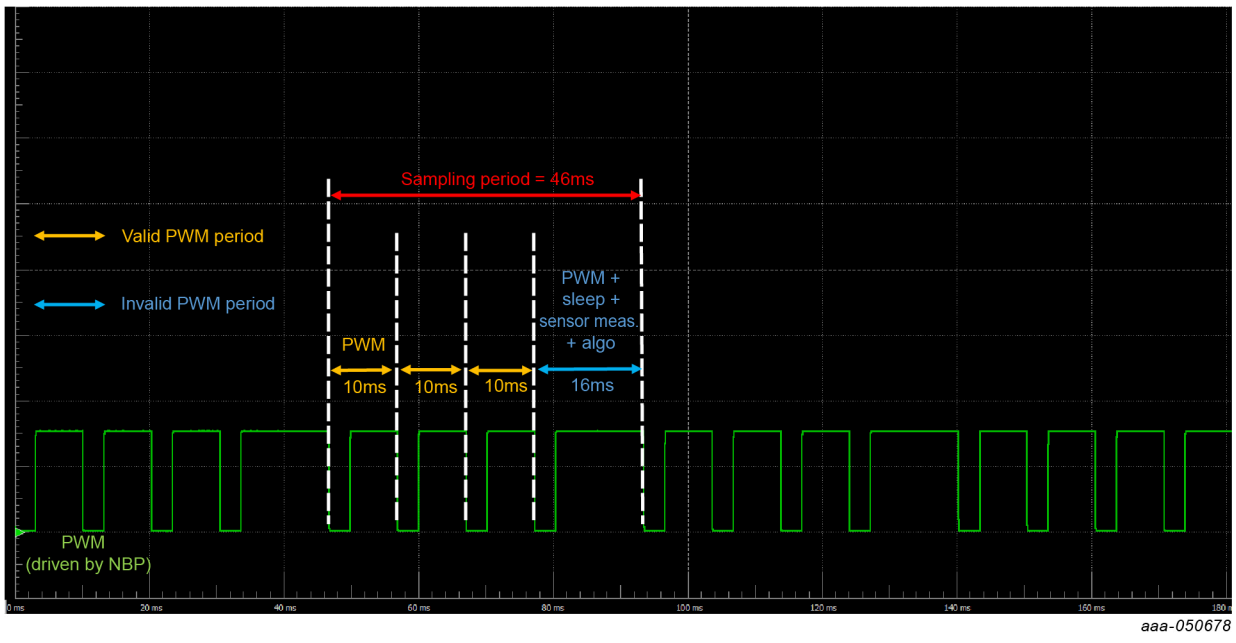


**Figure 14. PWM signal in Normal mode when self-test is not performed, with MODECFG_PWMPOL = 0**

# 9 Reading periodic pressure measurements from the NBP

Since the NBP runs its own pressure change algorithms at every sampling period, the Host does not typically need to periodically read pressure from the NBP. However, if periodic reading of the pressure value is needed by the Host, it can be accomplished using one of the following methods:

- NBP9 only: By decoding the PWM signal generated at every period in Normal mode. The PWM duty cycle encodes the latest pressure value read, or a specific status in case of error or warning.
- NBP8 and NBP9: By setting INTTRIG_SENSRDY bit. When SENSRDY bit is set, the NBP triggers a notification at every sampling period, after the periodic events have completed. The Host can therefore read the new pressure value at every period, following the notification from the NBP. If no error or warning occurred during the periodic events, only INTF bit of STATUS register is set when the NBP notifies the Host. The other bits in STATUS register are not set.
  An important point to be noted is that the duration of the notification and subsequent SPI transfers is not included in the sampling period configured. So, when SENSRDY bit is set, the resulting sampling period is equal to: *sampling period configured + duration of notification + duration of SPI transfers.*
  The duration of the notification corresponds to the duration of the INT pulse, plus the 200 ms delay before PS_ENABLE is asserted, if the PS_ENABLE feature is used. If PS_ENABLE feature is disabled, the 200 ms delay is not performed.
  For example, if the NBP8 sampling period configured is 10 ms, INT pulse duration is 4ms, PS_ENABLE feature is disabled and SPI transfers last less than 1 ms, then the resulting sampling period will be around 15 ms when self-test is not executed, and 18.5 ms when self-test is executed.
- NBP8 and NBP9: By the Host triggering a SPI communication request periodically. However, it is important to note that in such case, the period at which the Host triggers the request needs to be strictly greater than the NBP sampling period. If the Host triggers requests with a period shorter than the NBP sampling period, the NBP will not be able to take new measurements, so the pressure value and FIFO will not update.
  For such implementation, NXP recommends that the Host triggers SPI requests at a period twice longer than the NBP sampling period. For example, if the NBP8 sampling period configured is 10 ms, the Host should request SPI communication every 20 ms minimum.

***Note:***

*In the NBP8, the duration of the sampling period configured with PSP register only includes the duration of sensor measurements, execution of the pressure change algorithms and sleep. The duration of the self-test is not included.*

*In the NBP9, the duration of the sampling period configured with MODECFG_MODE bit includes the self-test, sensor measurements, execution of the pressure change algorithms, PWM generation in Normal mode, and sleep.*

# 10 Estimating the average current value

Table 4 and Table 5 provide information allowing to estimate the average current value depending on the sampling period and periodic self-test configurations. The values provided have been measured on one device at 25 °C, supplied with 3.3 V. There may be device-to-device variations, variations with supply voltage and temperature. The values are therefore provided as indication only.

**Table 4. NBP8 average current values measured on one device at 25 °C, 3.3 V**

| PSP configuration | Self-test performed? | Typical sampling period [ms] | Average current of one period [µA] |
|---|---|---|---|
| 0x00 *(typ. 10 ms)* | No | 10 | 582 |
| | Yes | 13.5 | 1083 |

**Table 4. NBP8 average current values measured on one device at 25 °C, 3.3 V*...continued***

| PSP configuration | Self-test performed? | Typical sampling period [ms] | Average current of one period [µA] |
|---|---|---|---|
| 0x01 *(typ. 20 ms)* | No | 20 | 278 |
| | Yes | 23.5 | 591 |
| 0x02 *(typ. 40 ms)* | No | 40 | 164 |
| | Yes | 43.5 | 368 |
| 0x03 *(typ. 70 ms)* | No | 70 | 90 |
| | Yes | 73.5 | 210 |
| 0x04 *(typ. 135 ms)* | No | 135 | 47 |
| | Yes | 138.5 | 113 |
| 0x05 *(typ. 510 ms)* | No | 510 | 13 |
| | Yes | 513.5 | 31 |
| 0x06 *(typ. 1 s)* | No | 1000 | 6.6 |
| | Yes | 1003.5 | 15 |

*Example:* STPER is configured to value 255 and PSP to value 0x04. The self-test is performed once every 255 sampling periods. The resulting average current value is equal to:

$$average\ \ current = \frac{254 \text{ x } 47 \text{ x } 135 + 113 \text{ x } 138.5}{254 \text{ x } 135 + 138.5} = 47.27\ \ uA \quad (1)$$

**Table 5. NBP9 average current values measured on one device at 25 °C, 3.3 V**

| MODECFG_MODE configuration | Self-test performed? | Typical sampling period [ms] | Average current of one period [µA] |
|---|---|---|---|
| 1 *(typ. 50 ms)* | No | 46.5 | 1900 |
| | Yes | 50 | 1940 |
| 0 *(typ. 500 ms)* | No | 496.5 | 13 |
| | Yes | 500 | 31 |

*Example:* STPER is configured to value 10 and MODECFG_MODE to value 0. The self-test is performed once every 10 sampling periods. The resulting average current value is equal to:

$$average\ \ current = \frac{9 \text{ x } 13 \text{ x } 496.5 + 31 \text{ x } 500}{9 \text{ x } 496.5 + 500} = 14.81\ \ uA \quad (2)$$

# 11 References

[1] NBP product webpage on NXP.com https://www.nxp.com/products/sensors/pressure-sensors/battery-pressure-monitoring-sensors-bpms/highly-integrated-battery-pressure-monitor-sensor:NBP8-9x?fpsp=1#design-resources

[2] NBP8 data sheet: https://www.nxp.com/docs/en/data-sheet/NBP8.pdf

[3] NBP9 data sheet: https://www.nxp.com/docs/en/data-sheet/NBP9.pdf

[4] Demo code for the Host connected to the NBP8: https://www.nxp.com/webapp/Download?colCode=NBP8x_Application&appType=license

[5] Demo code for the Host connected to the NBP9: https://www.nxp.com/webapp/Download?colCode=NBP9_demo_package&appType=license

[6] FXPS7xx0D4 product webpage
https://www.nxp.com/products/sensors/pressure-sensors/lpg-and-cng-gas-20-to-550-kpa/digital-absolute-pressure-sensor-20-to-550-kpa:FXPS7xx0D4

[7] FXPS7xx0A4 product webpage
https://www.nxp.com/products/sensors/pressure-sensors/lpg-and-cng-gas-20-to-550-kpa/analog-absolute-pressure-sensor-20-to-550-kpa:FXPS7xx0A4

# 12 Legal information

## 12.1 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.

## 12.2 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.

**Translations** — A non-English (translated) version of a document, including the legal information in that 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 vulnerabilities or may support established security standards or specifications with known limitations. 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.

## 12.3 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.

## Tables

Tab. 1. NBP default configuration after POR ....4
Tab. 2. Examples of SPI commands and responses following a notification from the NBP8 ....20
Tab. 3. Examples of SPI commands and responses following transfers requested by the Host ....20
Tab. 4. NBP8 average current values measured on one device at 25 °C, 3.3 V ....23
Tab. 5. NBP9 average current values measured on one device at 25 °C, 3.3 V ....24

## Figures

Fig. 1. Bit fields update during firmware verification ....9
Fig. 2. Bit fields update during Pcell self-test ....10
Fig. 3. Bit fields update during ADC self-test ....10
Fig. 4. Bit fields update during sensor measurements ....11
Fig. 5. Update strategies for STATUS and SENSTATUS bit fields ....12
Fig. 6. Input and output to the pressure compensation routine ....13
Fig. 7. Signals during an NBP8 transfer request ....14
Fig. 8. Signals during an NBP8 transfer request with PINCFG = 0x05 ....15
Fig. 9. Signals during an NBP9 transfer request ....16
Fig. 10. Signals during an NBP9 transfer request with PINCFG = 0x05 ....17
Fig. 11. Signals during transfers requested by the host ....19
Fig. 12. PWM generation following a notification, with MODECFG_PWMPOL = 0 ....21
Fig. 13. PWM signal in Normal mode when self-test is performed at every sampling period, with MODECFG_PWMPOL = 0 ....22
Fig. 14. PWM signal in Normal mode when self-test is not performed, with MODECFG_PWMPOL = 0 ....22

# Contents

**1 Introduction ..... 3**
1.1 Purpose ..... 3
1.2 Part number selection for battery pressure monitoring applications ..... 3
1.3 Configuration of the NBP parameters ..... 3
**2 NBP program start and default configuration ..... 4**
**3 Configuring the thresholds in the Pressure Change algorithms ..... 5**
3.1 General information ..... 5
3.2 Fixed Threshold algorithm ..... 5
3.3 Relative Threshold algorithm ..... 6
3.4 Slope Threshold algorithm ..... 7
**4 STATUS and SENSTATUS registers ..... 8**
4.1 Purpose ..... 8
4.2 Bit fields update ..... 9
4.3 Note on STATUS_INTF flag ..... 12
4.4 Handling errors ..... 12
**5 Handling an NBP transfer request ..... 13**
5.1 Handling a transfer request from the NBP8 ..... 13
5.2 Handling a transfer request from the NBP9 ..... 15
**6 Triggering a transfer request with the NBP8 and NBP9 ..... 17**
6.1 Preliminary check ..... 17
6.2 Flow ..... 18
6.3 Points of attention ..... 19
**7 SPI transfers ..... 20**
**8 Decoding the PWM with the NBP9 ..... 21**
**9 Reading periodic pressure measurements from the NBP ..... 23**
**10 Estimating the average current value ..... 23**
**11 References ..... 24**
**12 Legal information ..... 26**



For more information, please visit: http://www.nxp.com

**Date of release: 9 August 2023**
**Document identifier: AN13882**