UM12293

NPM8 Firmware Library User Manual

Rev. 1.0 — 25 July 2025 User manual

Document information

Information	Content
Keywords	IPMS, NPM8, Industrial Pressure Sensor
Abstract	This document describes the firmware library provided for all derivatives of the NPM8 family.

1 Introduction

This document describes the functions available in the NPM8 firmware Library.

The intended audience for this document includes firmware architects, developers, coders, and testers working with the NPM8 device.

This document is divided into three sections: this introduction, a section describing global variables, and standard formats used throughout the functions, and a third section describing each function.

2 Globals and formats

2.1 Global variables

Table 1 summarizes all global variables used by the NXP firmware library and their locations. Developers must account for these variables when creating user firmware.

Table 1. Global variable and their locations

Name	Address	Reference
IPMS_INTERRUPT_FLAG	8 Fh	Section 2.1.1

Note: *In the library, the global variable gu8TPMSLPDMFlag is declared at address 8 Eh. This variable is present for backward compatibility with previous library versions. The gu8TPMSLPDMFlag variable is not currently used by the NPM8 library functions, so users may use the address 8 Eh to store application variables.*

2.1.1 IPMS interrupt flag

This global variable tracks interrupts that have occurred. The NPM8 firmware Library uses it to track expected interrupts. Also, you can use this variable for your own purposes. Various firmware functions use the Stop1 or Stop4 modes. When the MCU enters Stop mode, the watchdog is halted. To provide a back-up recovery, users should use either the RTI or PWU which can be programmed for interrupt if a software or firmware routine has consumed too much time. The watchdog is automatically restarted when the program goes back in RUN mode.

The IPMS_INTERRUPT_FLAG is not cleared automatically. Users must clear this variable after power-on reset (POR).

Table 2 shows the IPMS_INTERRUPT_FLAG format. The trigger condition column describes conditions necessary for that flag to be set.

Table 2. IPMS_INTERRUPT_FLAG format and trigger conditions

Flag	Bit	Trigger condition
LVD Interrupt	7	LVD interrupt entered
PWU Interrupt	6	PWU interrupted entered
TOF Interrupt	5	TOF interrupt entered
LFR Error Interrupt	4	LFR is interrupted entered and the LFERF bit of the LFS register is set
ADC Interrupt	3	ADC interrupt entered
LFR Interrupt	2	LFR interrupt entered and the LFERF bit of the LFS register is clear
RTI Interrupt	1	RTI interrupted entered
KBI Interrupt	0	KBI interrupted entered

IPMS_INTERRUPT_FLAG is 1 byte long and is at address 8Fh. Users must account for this variable when developing for the NPM8.

2.2 Flash memory allocations

In the library, the functions and constant variables are declared under specific sections. In the PRM file of the application project, users can map these sections to the desired addresses in FLASH.

The library functions are declared under the sections FIRMWARE_SECTION_1, APP_LIB_SECTION and USER_ROM. The library constant variables are declared under the DEFAULT section, allocated to the ROM_VAR section in the PRM file of the application project.

2.3 Universal uncompensated measurement array (UUMA) format

The NPM8 measurement routines are divided into two subsets:

- Routines that return uncompensated measurements
- Routines that take uncompensated measurements as arguments and return compensated measurements

To be consistent and keep the number of CPU cycles down, all uncompensated measurement routines return data in the array format described in Table 3, and all compensating routines take data from the same array.

Table 3. Universal uncompensated measurement array

Index	Content
0	Uncompensated voltage
1	Uncompensated temperature
2	Uncompensated pressure
3	Uncompensated x-axis acceleration[1]
4	Uncompensated z-axis acceleration[2]

[1] Applicable to devices supporting x-axis measurement.
[2] Applicable to devices supporting z-axis measurement.

This array is referred to as Universal uncompensated measurement array (UUMA). It can be located anywhere the user decides.

Each element must be 16-bits long (2 bytes) regardless of what the actual bit-width of the measurement is.

Each individual uncompensated measurement routine only updates its corresponding item. For example, calling the IPMS_READ_VOLTAGE routine only modifies the voltage element of the array. The rest remains unchanged.

Compensation routines do not modify any elements in the UUMA.

2.4 Simulated SPI interface signal format

The NPM8 includes three routines (IPMS_MSG_INIT, IPMS_MSG_READ and IPMS_MSG_WRITE) that, when used together, allow the user to perform serial communication with the device through a simulated SPI interface.

The following assumptions are made:

- Only two pins are used: PTA0 for data (both incoming and outgoing) and PTA1 for clock. No follower select is included by default, but the user may use any other pin if required.
- The data pin has a pullup resistor enabled.
- The NPM8 is a central device (the NPM8 provides the clock).

- Data can be read/written 8 bits at a time.
- The speed of the interface depends on bus clock settings.
- Data is transferred to MSB first.
- A single line is used for both sending and receiving data (BIDIROE = SET according to NXP nomenclature).
  - At the rising edge of the clock, the central places data on the pin. It is valid unt il the falling edge of the clock. The follower must not drive the line during this period.
  - At the falling edge of the clock, the central makes the data pin an input and listens for data. The follower must then place data on the data line until the rising edge of the clock.
- Clock polarity = 0 (Normally low).
- Clock phase = 1 (First half is high).

Figure 1 shows the details of the simulated SPI interface.

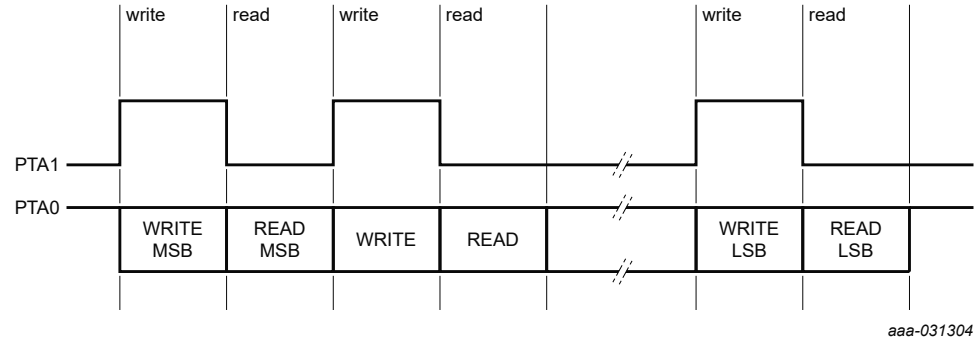

Figure 1. Description of the physical layer on the NPM8 simulated SPI interface

For further information on the use of the simulated SPI interface routines, see:

- Section 3.2.30 "Void IPMS_MSG_INIT (void)"
- Section 3.2.32 "UINT8 IPMS_MSG_READ (void)"
- Section 3.2.31 "UINT8 IPMS_MSG_WRITE (UINT8 u8SendByte)"

2.5 LFR registers initialized by firmware

Some LFR registers are touched by firmware when the function vnfLFRConfigFactoryRegs() provided in the NPM8 firmware library is executed. The goal of this action is to configure the LFR module in the best-known configuration for Manchester encoded reception.

LFR registers are configured differently, depending on the user-selected sensitivity. Table 4 and Table 5 describe these settings.

Note: *Shaded cells show the register touched by firmware; the loaded value is displayed.*

Table 4. Customer-configurable LF Register with SENS = 1

	Bit name							
Register name	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LFCTL1	LFEN	SRES	CARMOD	—	IDSEL		SENS	
LFCTL2	LFSTM				LFONTM			
LFCTL3	LFDO	TOGMOD	Synchronize		LFCDTM			
LFCTL4	LFDRIE	LFERIE	LFCDIE	LFIDIE	DECEN	VALEN	TIMOUT	
LFS	LFDRF	LFERF	LFCDF	LFIDF	LFOVF	LFEOMF	LPSM	LFIAK
LFDATA	RXDATA							
LFIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

Table 4. Customer-configurable LF Register with SENS = 1*...continued*

	Bit name							
Register name	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LFIDH	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
LFCTRLE	—	—	—	—	—	0	0	0
LFCTRLD	1	0	DEQS	1	1	1	0	0
LFCTRLC	0	0	0	1	AZEN	LOWQ		DEQEN
LFCTRLB	1	1	LFFAF	LFCAF	LFPOL	1	1	0
LFCTRLA	—	—	—	—	LFCC			
TRIM1	—	—	—	—	—	—	—	—
TRIM2	—	—	—	—	—	—	—	—

Note: *Shaded cells show the register touched by firmware; the loaded value is displayed.*

Table 5. Customer-configurable LF Register with SENS = 2

	Bit name							
Register name	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LFCTL1	LFEN	SRES	CARMOD	—	IDSEL		SENS	
LFCTL2	LFSTM				LFONTM			
LFCTL3	LFDO	TOGMOD	Synchronize		LFCDTM			
LFCTL4	LFDRIE	LFERIE	LFCDIE	LFIDIE	DECEN	VALEN	TIMOUT	
LFS	LFDRF	LFERF	LFCDF	LFIDF	LFOVF	LFEOMF	LPSM	LFIAK
LFDATA	RXDATA							
LFIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
LFIDH	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
LFCTRLE	—	—	—	—	—	0	0	0
LFCTRLD	1	0	DEQS	1	1	1	1	1
LFCTRLC	0	0	0	1	AZEN	LOWQ		DEQEN
LFCTRLB	1	1	LFFAF	LFCAF	LFPOL	1	1	0
LFCTRLA	—	—	—	—	LFCC			
TRIM1	—	—	—	—	—	—	—	—
TRIM2	—	—	—	—	—	—	—	—

3 Firmware functions

3.1 Firmware functions

Table 6 lists the available firmware functions for NPM8 single- and dual-axis accelerometers.

Table 6. NPM8 firmware functions

Return type	Function	Single/dual axis	Reference
VOID	IPMS_RESET	Single/dual	Section 3.2.1
UINT8	IPMS_READ_VOLTAGE	Single/dual	Section 3.2.2
UINT8	IPMS_COMP_VOLTAGE	Single/dual	Section 3.2.3
UINT8	IPMS_READ_TEMPERATURE	Single/dual	Section 3.2.4
UINT8	IPMS_COMP_TEMPERATURE	Single/dual	Section 3.2.5
UINT8	IPMS_READ_PRESSURE	Single/dual	Section 3.2.6
UINT8	IPMS_COMP_PRESSURE	Single/dual	Section 3.2.7
UINT8	IPMS_READ_ACCEL_X	Single-X/dual	Section 3.2.8
UINT8	IPMS_COMP_ACCEL_X	Single-X/dual	Section 3.2.9
UINT8	IPMS_READ_DYNAMIC_ACCEL_X	Single-X/dual	Section 3.2.10
UINT8	IPMS_READ_ACCEL_Z	Single-Z/dual	Section 3.2.11
UINT8	IPMS_COMP_ACCEL_Z	Single-Z/dual	Section 3.2.12
UINT8	IPMS_READ_DYNAMIC_ACCEL_Z	Single-Z/dual	Section 3.2.13
UINT8	IPMS_READ_V0	Single/dual	Section 3.2.14
UINT8	IPMS_READ_V1	Single/dual	Section 3.2.15
UINT8	IPMS_LFOCAL	Single/dual	Section 3.2.16
UINT8	IPMS_LFOCAL_BUSCLK	Single/dual	Section 3.2.17
UINT8	IPMS_MFOCAL	Single/dual	Section 3.2.18
UINT16	IPMS_WAVG	Single/dual	Section 3.2.19
VOID	IPMS_RF_ENABLE	Single/dual	Section 3.2.20
VOID	IPMS_RF_RESET	Single/dual	Section 3.2.21
VOID	IPMS_RF_READ_DATA	Single/dual	Section 3.2.22
VOID	IPMS_RF_READ_DATA_REVERSE	Single/dual	Section 3.2.23
VOID	IPMS_RF_WRITE_DATA	Single/dual	Section 3.2.24
VOID	IPMS_RF_WRITE_DATA_REVERSE	Single/dual	Section 3.2.25
VOID	IPMS_RF_CONFIG_DATA	Single/dual	Section 3.2.26
VOID	IPMS_RF_SET_TX	Single/dual	Section 3.2.27
VOID	IPMS_READ_ID	Single/dual	Section 3.2.28
VOID	IPMS_RF_DYNAMIC_POWER	Single/dual	Section 3.2.29
VOID	IPMS_MSG_INIT	Single/dual	Section 3.2.30
UINT8	IPMS_MSG_WRITE	Single/dual	Section 3.2.31
UINT8	IPMS_MSG_READ	Single/dual	Section 3.2.32

Table 6. NPM8 firmware functions*...continued*

Return type	Function	Single/dual axis	Reference
UINT8	IPMS_CHECKSUM_XOR	Single/dual	Section 3.2.33
UINT8	IPMS_CRC8	Single/dual	Section 3.2.34
UINT16	IPMS_SQUARE_ROOT	Single/dual	Section 3.2.35
VOID	IPMS_LF_ENABLE	Single/dual	Section 3.2.36
UINT8	IPMS_LF_READ_DATA	Single/dual	Section 3.2.37
UINT8	IPMS_WIRE_AND_ADC_CHECK	Single/dual	Section 3.2.38
VOID	IPMS_FLASH_WRITE	Single/dual	Section 3.2.39
UINT16	IPMS_FLASH_ERASE	Single/dual	Section 3.2.40
VOID	IPMS_MULT_SIGN_INT16	Single/dual	Section 3.2.41
UINT8	IPMS_VREG_CHECK	Single/dual	Section 3.2.42
UINT8	IPMS_E_READ_ACCEL_X	Single-X/dual	Section 3.2.43
UINT8	IPMS_E_READ_PRESSURE	Single/dual	Section 3.2.44
UINT8	IPMS_E_READ_ACCEL_Z	Single-Z/dual	Section 3.2.45
VOID	IPMS_E_FRC_ENABLE	Single/dual	Section 3.2.46
VOID	IPMS_E_FRC_DISABLE	Single/dual	Section 3.2.47
VOID	IPMS_E_FRC_CLEAR	Single/dual	Section 3.2.48
VOID	IPMS_E_FRC_READ	Single/dual	Section 3.2.49
UINT8	IPMS_E_FRC_CALIB	Single/dual	Section 3.2.50
UINT8	IPMS_E_FRC_CALIB_BUSCLK	Single/dual	Section 3.2.51
UINT8	IPMS_PRECHARGE_EN	Single/dual	Section 3.2.52
VOID	IPMS_E_ACTIVATE_CHANNEL	Single/dual	Section 3.2.53
VOID	IPMS_E_DEACTIVATE_CHANNEL	Single/dual	Section 3.2.54
UINT8	IPMS_E_READ_CONT_ACCEL_X	Single-X/dual	Section 3.2.55
UINT8	IPMS_E_READ_CONT_ACCEL_Z	Single-Z/dual	Section 3.2.56
UINT8	IPMS_E_READ_CONT_PRESSURE	Single/dual	Section 3.2.57
UINT8	IPMS_E_READ_DYNAMIC_ACCEL_Z	Single-Z/dual	Section 3.2.58
UINT8	IPMS_E_READ_DYNAMIC_ACCEL_X	Single-X/dual	Section 3.2.59

3.2 Function description

The following function descriptions include stack sizes and approximate duration.

Stack sizes have been calculated by executing each routine and measuring the amount of memory used. Unless noted, these sizes represent the maximum stack the function uses.

Duration estimates are performed on one part at room temperature. These estimates are intended to serve as a guideline for typical execution time.

Table 7 and Table 8 describe the SMI initial and subsequent delays.

Table 7. SMI initial delay

SMI initial delay	
SAMP_DLY_INIT_104US	0x0h
SAMP_DLY_INIT_128US	0x1h
SAMP_DLY_INIT_160US	0x2h
SAMP_DLY_INIT_200US	0x3h
SAMP_DLY_INIT_256US	0x4h
SAMP_DLY_INIT_320US	0x5h
SAMP_DLY_INIT_408US	0x6h
SAMP_DLY_INIT_512US	0x7h
SAMP_DLY_INIT_648US	0x8h
SAMP_DLY_INIT_816US	0x9h
SAMP_DLY_INIT_1024US	0xAh
SAMP_DLY_INIT_1288US	0xBh
SAMP_DLY_INIT_1624US	0xCh
SAMP_DLY_INIT_2048US	0xDh
SAMP_DLY_INIT_2584US	0xEh
SAMP_DLY_INIT_3248US	0xFh

Table 8. SMI subsequent delay

SMI subsequent delay	
SAMP_DLY_SUBS_64US	0x00h
SAMP_DLY_SUBS_80US	0x10h
SAMP_DLY_SUBS_104US	0x20h
SAMP_DLY_SUBS_128US	0x30h
SAMP_DLY_SUBS_160US	0x40h
SAMP_DLY_SUBS_200US	0x50h
SAMP_DLY_SUBS_256US	0x60h
SAMP_DLY_SUBS_320US	0x70h
SAMP_DLY_SUBS_408US	0x80h
SAMP_DLY_SUBS_512US	0x90h
SAMP_DLY_SUBS_648US	0xA0h
SAMP_DLY_SUBS_816US	0xB0h
SAMP_DLY_SUBS_1024US	0xC0h
SAMP_DLY_SUBS_1288US	0xD0h
SAMP_DLY_SUBS_1624US	0xE0h
SAMP_DLY_SUBS_2048US	0xF0h

3.2.1 Void_IPMS_RESET (void)

- **Description:** This function calls the function vnfLFRConfigFactoryRegs() described in Section 2.5. Next, it resets the stack pointer to the last RAM location and jumps to the main() function of the user application project. No further initialization is performed.
- **Stack size:** 2 bytes
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await interruptions. It is not affected by interruptions either.
- **Resources:** Stack
- **Input parameters:**
  - None
- **Returns:**
  - Void

3.2.2 UINT8 IPMS_READ_VOLTAGE (UINT16 *u16UUMA)

- **Description:** Performs a 10-bit uncompensated voltage measurement and places it in the UUMA. While waiting for the ADC to converge, this function goes into STOP4. If the ADC, for an unexpected reason, fails to converge, this function has the following built-in timeout: After five continuous non-ADC interrupts, the function will assume a failed ADC reading, flag it so, and exit.
  - If the ADC value is over or under the normal operating condition, the voltage error status flag is set. The expected voltage result is forced to either 0 V or 1023 V.
  - If the ADC times out with no result, the ADC error status flag is set.
  - Measurements below 2.1 V are not guaranteed for accuracy.
- **Stack size:** 23 bytes
- **Approximate duration:** 99 µs
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** ADC, band gap
- **Input parameters:**
  - UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3). Only the 10-bit uncompensated voltage result is updated.
- **Returns:**
  - UINT8 u8Status: Valid error flags/outputs are described in Table 9.

Table 9. Valid output conditions for IPMS_READ_VOLTAGE

u8Status Value	Measurement value	Condition
20h	03FFh	Uncompensated voltage reading outside the valid range (high)
20h	0000h	Uncompensated voltage reading outside the valid range (low)
80h	Undefined	Uncompensated voltage reading not acquired
00h	0001h to 03FEh	Valid uncompensated voltage reading

Note: *The band gap bit (bit 0 in the SPMSC1 register) must be set prior to calling this function for results to be valid.*

3.2.3 UINT8 IPMS_COMP_VOLTAGE (UINT8 *u8CompVoltage, *UINT16 u16UUMA)

- **Description:** Performs an 8-bit compensated voltage measurement. It is the responsibility of the user to ensure that an updated and valid uncompensated voltage reading is available in the UUMA for this routine to return a meaningful value.
  - If V_{out} < 1.7 V, the result is 1 and the over/underflow status flag will be set.
  - Measurements below 2.1 V are not guaranteed for accuracy.
  - If V_{out} ≥ 3.7 V, the result is FFh and the over/underflow status flag will be set.
  - For repeatability data, refer to the NPM8 family data sheets.
- **Stack size:** 31 bytes
- **Approximate duration:** 204 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await interrupts. It is not affected by interrupts either.
- **Resources:** UUMA
- **Input parameters:**
  - UINT8 *u8CompVoltage: Updated 8-bit compensated voltage result.
  - UINT16 *u16UUMA: Pointer to UUMA(as described in Section 2.3). Uncompensated voltage is utilized from this array.
- **Returns:**
  - UINT8 u8Status: Valid error flags/outputs are described in Table 10.

Table 10. Valid output conditions for IPMS_COMP_VOLTAGE

u8Status Value	Measurement value	Condition
01h	FFh	Compensated voltage reading outside the valid range (high)
01h	01h	Compensated voltage reading outside the valid range (low)
00h	02h to FEh	Valid compensated voltage reading

3.2.4 UINT8 IPMS_READ_TEMPERATURE (UINT16 *u16UUMA)

- **Description:** Performs a 12-bit uncompensated temperature measurement and places the result in the UUMA. While waiting for the ADC to converge, this function goes into STOP4. If the ADC, for an unexpected reason, fails to converge, this function has the following built-in timeout: After five continuous non-ADC interrupts, the function will assume a failed ADC reading, flag it so, and exit. If the LVWF hardware bit is set, it flags it accordingly as well. The LVWF flag is cleared before the function returns.
  - If the ADC value is over or under the normal operating condition, the temperature error status flag is set. The expected temperature result is forced to either 0 or 4095.
  - If the ADC times out with no result, the ADC error status flag is set.
- **Stack size:** 18 bytes
- **Approximate duration:** 219 µs
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** ADC, band gap
- **Input parameters:**
  - UINT16 *u16UUMA: Pointer to UMMA (as described in Section 2.3). Only the 12-bit uncompensated temperature result is updated.
- **Returns:**
  - UINT8 u8Status:Valid error flags/outputs are described in Table 11.

Table 11. Valid output conditions for IPMS_READ_TEMPERATURE

u8Status Value	Measurement value	Condition
40h	0FFFh	Uncompensated temperature reading outside the valid range (high)
40h	0000h	Uncompensated temperature reading outside the valid range (low)
60h	0FFFh	Uncompensated temperature reading outside the valid range (high) and LVWF set
60h	0000h	Uncompensated temperature reading outside the valid range (low) and LVWF set
80h	Undefined	Uncompensated temperature reading not acquired
A0h	Undefined	Uncompensated temperature reading not acquired and LVWF set
00h	0001h to 0FFEh	Valid uncompensated temperature reading
20 h	0001h to 0FFEh	Valid uncompensated temperature reading and LVWF set

Note: *The band gap bit (bit 0 in the SPMSC1 register) must be set prior to calling this function for results to be valid.*

3.2.5 UINT8 IPMS_COMP_TEMPERATURE (UINT8 *u8Temp, UINT16 *u16UUMA)

- **Description:** Performs an 8-bit compensated temperature measurement. It is the responsibility of the user to ensure that an updated and valid uncompensated temperature reading is available in the UUMA for this routine to return a meaningful value.
  - If T_{out} < -50 °C, u8Temp is 1 and the over/underflow status flag is set.
  - If T_{out} ≥ 200 °C, u8Temp is FFh and the over/underflow status flag is set.
- **Stack size:** 30 bytes
- **Approximate duration:** 209 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await interrupts. It is not affected by interrupts either.
- **Resources:** UUMA
- **Input parameters:**
  - UINT8 *u8Temp: Updated 8-bit compensated temperature result.
  - UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3). Uncompensated temperature is used from this array.
- **Returns:** UINT8 u8Status: Valid error flags/outputs are described in Table 12.

Table 12. Valid output conditions for IPMS_COMP_TEMPERATURE

u8Status Value	Measurement value	Condition
01h	FFh	Compensated temperature reading outside the valid range (high)
01h	01h	Compensated temperature reading outside the valid range (low)
00 h	02h to FEh	Valid compensated temperature reading

3.2.6 UINT8 IPMS_READ_PRESSURE (UINT16 *u16UUMA, UINT8 u8Avg)

- **Description:** Performs a 12-bit uncompensated pressure measurement and places it in the UUMA. The function configures an initial delay of 2048 µs and a subsequent delay of 512 µs. While waiting for the ADC to converge, this function goes into STOP4. If the ADC, for an unexpected reason, fails to converge, this function has the following built-in timeout: After five continuous non-ADC interrupts, the function will assume a failed ADC reading, flag it so, and exit. If the LVWF hardware bit is set, it will flag it accordingly as well. The LVWF flag is cleared before the function returns.
  - If the ADC value is over or under the normal operating condition, the pressure error status flag is set. The expected pressure result is forced to either 0 or 4095.
  - If the ADC times out with no result, the ADC error status flag is set.
- **Stack size:** 32 bytes
- **Approximate duration:**

Table 13. Approximate duration for IPMS_READ_PRESSURE

Mode	Component	Estimated duration during normal operation	Observed duration [ms]
Average of 1	Bus clock cycles	840	0.21
	Initial delay MFO clock cycles	256	2.04
	ADC conversion time [µs] [1]	20	0.02
	STOP4 exit time [µs] [2]	114	0.014
	Total [ms]	—	2.284
Average of 4	More bus clock cycles per more sample	36	0.009
	Subsequent MFO clock cycles per more sample	64	0.512
	More ADC conversion time per more sample [1]	20	0.02
	More STOP4 exit time per sample [2]	14	0.014
	More time per more sample excludes subsequent delay [ms]	0.043	—
	Total for average = 4 [ms]	—	3.949

[1] Typical ADC conversion time with nominal ADC clock at conversion settings.
[2] Typical STOP4 exit time. For an exact range, refer to the product data sheet.

- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** SMI, ADC, internal bond wires
- **Input parameters:**
  - UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3). Only the 12-bit uncompensated pressure result is updated.
  - UINT8 u8Avg: The number of measurements to average into one result. The value can be set to 1, 2, 4, 8, or 16.
- **Returns:** UINT8 u8Status: Valid error flags/outputs are described in Table 14.

Table 14. Valid output conditions for IPMS_READ_PRESSURE

u8Status Value	Measurement value	Condition
04h	0FFF h	Uncompensated pressure reading outside the valid range (high)
04h	0000 h	Uncompensated pressure reading outside the valid range (low)
24h	0FFF h	Uncompensated pressure reading outside the valid range (high), and LVWF set
24h	0000 h	Uncompensated pressure reading outside the valid range (low), and LVWF set
80h	0000 h	Uncompensated pressure reading not acquired
A0h	0000 h	Uncompensated pressure reading not acquired, and LVWF set
00h	0001h to 0FFE h	Valid uncompensated pressure reading
20 h	0001h to 0FFE h	Valid uncompensated pressure reading, and LVWF set

3.2.7 UINT8 IPMS_COMP_PRESSURE (UINT16 *u16CompPressure, UINT16 *u16UUMA)

- **Description:** Performs a 10-bit compensated pressure measurement. It is the responsibility of the user to ensure that updated and valid uncompensated voltage, temperature, and pressure readings are available in the UUMA for this routine to return a meaningful value.
  - If either the temperature or the supply voltage measurements, result in a fault, inherent to this function, the pressure reading is forced to 0 and the appropriate pressure, temperature, and/or voltage flags are set in the status flag.
  - If P_{out} < 90 kPa, the over/underflow status flag is set, and u16CompPressure is forced to 001 h.
  - If P_{out} ≥ maximum pressure for the part number, u16CompPressure will be 3FF h and the over/underflow status flag is set.
  - If the passed uncompensated voltage measurement is estimated to be under the guaranteed operational region, the routine sets the voltage status flag. The accuracy of the returned value is not guaranteed.
- **Stack size:** 38 bytes
- **Approximate duration:** 766.5 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await interrupts. It is not affected by interrupts either.
- **Resources:** UUMA
- **In-put parameters:**
  - u16CompPressure: Updated 10-bit compensated pressure result.
  - UINT16 *u16UUMA: Pointer to UUMA uncompensated (as described in Section 2.3). Uncompensated voltage, temperature, and pressure are taken from this array.
- **Returns:** UINT8 u8Status: Valid error flags/outputs are described in Table 15.

Table 15. Valid output conditions for IPMS_COMP_PRESSURE

u8Status Value	Measurement value	Condition
01h	03FF h	Compensated pressure reading outside the valid range (high).
01h	0001 h	Compensated pressure reading outside the valid range (low).
21h	03FF h	Compensated pressure reading outside the valid range (high), and uncompensated voltage suspected to be below the valid operating range for this function.

Table 15. Valid output conditions for IPMS_COMP_PRESSURE...*continued*

u8Status Value	Measurement value	Condition
21h	0001 h	Compensated pressure reading outside the valid range (low), and uncompensated voltage suspected to be under below the operating range for this function.
20h	0002h to 03FE h	Uncompensated voltage is suspected to be below the valid operating range for this function. The compensated reading is not guaranteed for accuracy.
00h	0002h to 03FE h	Valid compensated pressure reading.

3.2.8 UINT8 IPMS_READ_ACCEL_X

UINT16 *u16UUMA, UINT8 u8Avg, UINT8 u8Filter, UINT8 u8OffsetIndex

- **Description:** It performs an uncompensated 12-bit measurement for the x-axis. The function configures an initial delay of 2048 µs and a subsequent delay of 512 µs. While waiting for the ADC to converge, this function goes into STOP4. If the ADC, for an unexpected reason, fails to converge, this function has the following built-in timeout: After five continuous non-ADC interrupts, the function will assume a failed ADC reading, flag it so, and exit. If the LVWF hardware bit is set, it flags it accordingly as well. The LVWF flag is cleared before the function returns.
  - If the ADC value is over or under the normal operating condition, the acceleration error status flag is set. The expected acceleration result is forced to either 0 or 4095.
  - If the ADC times out with no result, the ADC error status flag is set.
- **Stack size:** 37 bytes
- **Approximate duration:**

Table 16. Approximate duration for IPMS_READ_ACCEL_X

Mode	Component	Estimated duration during normal operation	Observed duration [ms]
500 Hz, Average of 1	Bus clock cycles	836	0.209
	Initial delay MFO clock cycles	256	2.04
	ADC conversion time [µs] [1]	20	0.02
	STOP4 exit time [µs] [2]	14	0.014
	Total [ms]	—	2.283
500 Hz, Average of 4	More bus clock cycles per more sample	38	0.0096
	Subsequent MFO clock cycles per more sample	64	0.512
	More ADC conversion time per more sample [1]	20	0.02
	More STOP4 exit time per sample [2]	14	0.014
	More time per more sample [ms]	0.0436	—
	Total for average = 4 [ms]	—	3.9498

[1] Typical ADC conversion time with nominal ADC clock at conversion settings.
[2] Typical STOP4 exit time. For an exact range, refer to the product data sheet.

- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from Stop mode.

- **Resources:** SMI, ADC, internal bond wires
- **Input parameters:**
  - UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3). Only the 12-bit uncompensated acceleration result is updated.
  - UINT8 u8Avg: The number of measurements to average into one result. The value can be set to 1, 2, 4, 8, or 16.
  - UINT8 u8Filter: Is set up the acceleration measurement based on Table 17.

Table 17. u8Filter options

u8 Filter Value	Selected filter
0	250 Hz low-pass filter selected
1	500 Hz low-pass filter selected
2	1000 Hz low-pass filter selected
3	2000 Hz low-pass filter selected

- UINT8 u8OffsetIndex: Selects the offset setting for the appropriate acceleration reading. The default index is 7. The valid range is 0 to 15.
- **Returns:** UINT8 u8Status: Valid error flags/outputs are described in Table 18.

Table 18. Valid output conditions for IPMS_READ_ACCEL_X

u8Status value	Measurement value	Condition
08h	0FFFh	Uncompensated acceleration reading outside the valid range (high).
08h	0000h	Uncompensated acceleration reading outside the valid range (low).
28h	0FFFh	Uncompensated acceleration reading outside the valid range (high), and LVWF set.
28h	0000h	Uncompensated acceleration reading outside the valid range (low), and LVWF set.
80h	0000h	Uncompensated acceleration reading is not acquired.
A0h	0000h	Uncompensated acceleration reading not acquired, and LVWF set.
00h	0001h to 0FFEh	Valid uncompensated acceleration reading.
20h	0001h to 0FFEh	Valid uncompensated acceleration reading, but LVWF set.

3.2.9 UINT8 IPMS_COMP_ACCEL_X (UINT16 *u16CompAccel, UINT16* u16UUMA)

- **Description:** Performs a 10-bit compensated acceleration measurement for the x-axis. It is the responsibility of the user to ensure that updated and valid uncompensated voltage, temperature, and acceleration readings are available in the UUMA for this routine to return a meaningful value.
  - If u16CompAccel rails low, u16CompAccel forces to 1 and the over/underflow status flag IS set.
  - If u16CompAccel rails high, u16CompAccel forces to 3FFh and the over/underflow status flag are is set.
  - If the incoming uncompensated voltage measurement is estimated to be under the guaranteed operational region, the routine sets the Voltage status flag. The accuracy of the returned value is not guaranteed.
  - For repeatability data, refer to the NPM8 family data sheets.
- **Stack size:** 48 bytes
- **Approximate duration:** 843 µs

- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await interrupts. It is not affected by interrupts either.
- **Resources:** UUMA
- **Input parameters:**
  - UINT16 *u16CompAccel: Updated 10-bit compensated acceleration.
  - UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3). Uncompensated voltage, temperature, and acceleration are taken from this array.
- **Returns:**
  - UINT8 u8Status: Valid error flags/outputs are described in Table 19.

Table 19. Valid output conditions for IPMS_COMP_ACCEL_X

u8Status Value	Measurement value	Condition
01h	03FFh	Compensated acceleration reading outside the valid range (high)
01h	0001h	Compensated acceleration reading outside the valid range (low)
21h	03FFh	Compensated acceleration reading outside the valid range (high), and uncompensated voltage suspected to be below the valid operating range for this function
21h	0001h	Compensated acceleration reading outside the valid range (low), and uncompensated voltage suspected to be below the valid operating range for this function
20h	0002h to 03FEh	Uncompensated voltage is suspected to be below the valid operating range for this function; The compensated reading is not guaranteed for accuracy
00h	0002h to 03FEh	Valid compensated acceleration reading

3.2.10 UINT8 IPMS_READ_DYNAMIC_ACCEL_X (UINT8 u8Filter, UINT8* u8Offset, UINT16* u16UUMA)

- **Description:** This function automatically executes a IPMS_READ_ACCEL_X measurement with a given initial dynamic offset. If the result is too high or too low, it changes the dynamic offset value and reexecute IPMS_READ_ACCEL_X until a) the result is valid or b). Then the result is railed high or low, and there are no more offset steps. Offset and uncompensated acceleration inside the UUMA are updated. The function configures an initial delay of 2048 µs.
- **Stack size:** 48 bytes.
- **Approximate duration:** 2950 µs when the starting offset is in target; 29,050 µs when the offset is 10 steps away.
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** SMI, ADC, internal bond wires.
- **Input parameters:**
  - UINT8 u8Filter: If nonzero, 250 Hz filter enabled. Otherwise, a 500 Hz filter selected.
  - UINT8* u8Offset: Pointer to initial offset step to load. Valid offset steps range from 0 to 15 and are described in the devices data sheet. An updated offset value is returned at the end of the function. In the case the acceleration is too high or too low and the function has run out of offset steps, a value of 255 (“0 to 1“) or 16 (”15 + 1”) shall be returned.

– UINT16 *u16UUMA: Pointer to the UUMA. Uncompensated acceleration is updated accordingly.

- **Returns:**
  – UINT8 u8Status: See Section 3.2.8, IPMS_READ_ACCEL_X for more information on the format of this status byte.

3.2.11 UINT8 IPMS_READ_ACCEL_Z (UINT16 *u16UUMA, UINT8 u8Avg, UINT8 u8Filter, UINT8 u8OffsetIndex)

- **Description:** Performs an uncompensated 12-bit measurement. The function configures an initial delay of 2048 us and a subsequent delay of 512 µs. While waiting for the ADC to converge, this function goes into STOP4. If the ADC, for an unexpected reason, fails to converge, this function has the following built-in timeout: After five continuous non-ADC interrupts, the function will assume a failed ADC reading, flag it so, and exit. If the LVWF hardware bit is set, it flags it accordingly as well. The LVWF flag is cleared before the function returns.
  – If the ADC value is over or under the normal operating condition, the acceleration error status flag is set. The expected acceleration result is forced to either 0 (rail high) or 4095 (rail low).
  – If the ADC times out with no result, the ADC error status flag is set.
- **Stack size:** 37 bytes
- **Approximate duration:**

Table 20. Approximate duration for IPMS_READ_ACCEL_Z

Mode	Component	Estimated duration during normal operation	Observed duration [ms]
500 Hz, Average of 1	Bus clock cycles	836	0.209
	Initial delay MFO clock cycles	256	2.04
	ADC conversion time [µs] [1]	20	0.02
	STOP4 exit time [µs] [2]	14	0.014
	Total [ms]	—	2.283
500 Hz, Average of 4	More bus clock cycles per more sample	38	0.0096
	Subsequent MFO clock cycles per more sample	64	0.512
	More ADC conversion time per more sample [1]	20	0.02
	More STOP4 exit time per more sample [2]	14	0.014
	More time per more sample [ms]	0.0436	—
	Total for average = 4 [ms]	—	3.9498

[1] Typical ADC conversion time with nominal ADC clock at conversion settings.
[2] Typical STOP4 exit time. For an exact range, refer to the product data sheet.

- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** SMI, ADC, internal bond wires
- **Input parameters:**
  – UINT16 *u16UUMA: Pointer to UUMA(as described in Section 2.3). Only the 12-bit uncompensated acceleration result is updated.
  – UINT8 u8Avg: The number of measurements to average into one result. The value can be set to 1, 2, 4, 8, or 16.

– UINT8 u8Filter: Will set up the acceleration measurement based on Table 21.
– UINT8 u8OffsetIndex: Selects the offset setting for the appropriate acceleration reading. The default index is 7. The valid range is 0 to 15.
– **Returns:** UINT8 u8Status: Valid error flags/outputs are described in Table 22.

Table 21. u8 Filter options

u8 Filter Value	Selected filter
0	250 Hz low-pass filter selected
1	500 Hz low-pass filter selected
2	1000 Hz low-pass filter selected
3	2000 Hz low-pass filter selected

Table 22. Valid output conditions for IPMS_READ_ACCEL_Z

u8Status Value	Measurement value	Condition
10h	0FFFh	Uncompensated acceleration reading outside the valid range (high).
10h	0000h	Uncompensated acceleration reading outside the valid range (low).
30h	0FFFh	Uncompensated acceleration reading outside the valid range (high), and LVWF set.
30h	0000h	Uncompensated acceleration reading outside the valid range (low), and LVWF set.
80h	0000h	Uncompensated acceleration reading is not acquired.
A0h	0000h	Uncompensated acceleration reading not acquired, and LVWF set.
00h	0001h to 0FFEh	Valid uncompensated acceleration reading.
20h	0001h to 0FFEh	Valid uncompensated acceleration reading, but LVWF set.

3.2.12 UINT8 IPMS_COMP_ACCEL_Z (UINT16 *u16CompAccel, UINT16* u16UUMA)

- **Description:** Performs a 10-bit compensated acceleration measurement. It is the responsibility of the user to ensure that updated and valid uncompensated voltage, temperature, and acceleration readings are available in the UUMA for this routine to return a meaningful value.
  – If u16CompAccel rails low, u16CompAccel is forced to 1 and the over/underflow status flag is set.
  – If u16CompAccel rails high, u16CompAccel is forced to 3FF h and the over/underflow status flag is set.
  – If the incoming uncompensated voltage measurement is estimated to be under the guaranteed operational region, the routine sets the voltage status flag. The accuracy of the returned value is not guaranteed.
  – For repeatability data, refer to the NPM8 family data sheets.
- **Stack size:** 48 bytes
- **Approximate duration:** 843 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await interruptions. It is not affected by interruptions either.
- **Resources:** UUMA
- **Input parameters:**
  – UINT16 *u16Accel: Updated 10-bit compensated acceleration.

– UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3). Uncompensated voltage, temperature, and acceleration will be taken from this array.

- **Returns:**
  – UINT8 u8Status: Valid error flags/outputs are described in Table 23.

Table 23. Valid output conditions for IPMS_COMP_ACCEL_Z

u8Status Value	Measurement value	Condition
01h	03FFh	Compensated acceleration reading outside the valid range (high)
01h	0001h	Compensated acceleration reading outside the valid range (low)
21h	03FFh	Compensated acceleration reading outside the valid range (high), and uncompensated voltage suspected to be below the valid operating range for this function
21h	0001h	Compensated acceleration reading outside the valid range (low), and uncompensated voltage suspected to be below the valid operating range for this function
20h	0002h to 03FEh	Uncompensated voltage is suspected to be below the valid operating range for this function; The compensated reading is not guaranteed for accuracy
00h	0002h to 03FEh	Valid compensated acceleration reading

3.2.13 UINT8 IPMS_READ_DYNAMIC_ACCEL_Z (UINT8 u8Filter, UINT8* u8Offset, UINT16* u16UUMA)

- **Description:** This function automatically executes a IPMS_READ_ACCEL_Z measurement with a given initial dynamic offset. If the result is too high or too low, it changes the dynamic offset value and reexecute IPMS_READ_ACCEL_Z until a) the result is valid or b) the result is railed high or low and there are no more offset steps. Offset and uncompensated acceleration inside the UUMA are updated. The function configures an initial delay of 2048 µs.
- **Stack size:** 48 bytes
- **Approximate duration:** 2950 µs when the starting offset is in target; 29,050 µs when the offset is 10 steps away.
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** SMI, ADC, internal bond wires.
- **Input parameters:**
  – UINT8 u8Filter: If nonzero, 250 Hz filter enabled. Otherwise, a 500 Hz filter selected.
  – UINT8* u8Offset: Pointer to initial offset step to load. Valid offset steps range from 0 to 15 and are described in the devices data sheet. An updated offset value is returned at the end of the function. In the case the acceleration is too high or too low and the function has run out of offset steps, a value of 255 ("0 to 1“) or 16 (”15 + 1") shall be returned.
  – UINT16 *u16UUMA: Pointer to the UUMA. Uncompensated acceleration is updated accordingly.
- **Returns:**
  – UINT8 u8Status: See Section 3.2.11 for more information on the format of this status byte.

3.2.14 UINT8 IPMS_READ_V0 (UINT16 *u16Result, UINT8 u8Avg)

- **Description:** Performs a 10-bit uncompensated measurement at pin PTB0.

- **Stack size:** 24 bytes
- **Approximate duration:** 102 µs
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** ADC, PTB0.
- **Input parameters:**
  - UINT16 *u16Result: Updated 10-bit uncompensated measurement.
  - UINT8 u8Avg: The number of measurements to average into one result. The value can be set to 1, 2, 4, 8, or 16.
- **Returns:**
  - UINT8 u8Status: Valid error flags/outputs are described in Table 24.

Table 24. Valid output conditions for IPMS_READ_V0 and IPMS_READ_V1

u8Status Value	Measurement value	Condition
01h	0000h	Reading not acquired
00h	Between 0000h - 03FFh	Valid reading

3.2.15 UINT8 IPMS_READ_V1 (UINT16 *u16Result, UINT8 u8Avg)

- **Description:** Performs a 10-bit uncompensated measurement at pin PTB1.
- **Stack size:** 24 bytes
- **Approximate duration:** 103 µs
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** ADC, PTB1.
- **Input parameters:**
  - UINT16 *u16Result: Updated 10-bit uncompensated measurement.
  - UINT8 u8Avg: The number of measurements to average into one result. The value can be set to 1, 2, 4, 8, or 16.
- **Returns:**
  - UINT8 u8Status: Valid error flags/outputs are described in Table 24.

3.2.16 UINT8 IPMS_LFOCAL (void)

- **Description:** Performs PWU clock calibration. The wake-up and periodic reset time can be calibrated more accurately by using the IPMS_LFOCAL firmware subroutine. This subroutine turns on the RFM crystal oscillator and feeds a 500 kHz clock via the DX signal to the TPM1 for one cycle of the LFO, but first executes a test to verify the presence of the external XTAL. The measured time is used to calculate the correct value for the WDIV0:5 bits for a WCLK period of 1 second. The resulting value for use in the WDIV0:5 bits is returned by the function. The user can decide whether to load the value to the WDIV0:5 bits or store for future reference. If the returned value is out-of-range, for example when the LFO is out of spec, the returned value is truncated to the minimum or the maximum (0h or 3Fh). The IPMS_LFOCAL subroutine cannot be used while the RFM is transmitting or the TPM1 is being used for another task. This routine will also consume more power due to the crystal oscillator running. This function accesses and writes data to the SIMOPT2 register. Because some of the bits in this register are write-once-only, it should be configured prior to calling this routine.
- **Stack size:** 11 bytes
- **Approximate duration:** 1580 µs

- **Power management:** This function executes entirely in RUN mode. It requires the MCU to be configured for a 4 MHz bus clock, and the RFM to be enabled but not transmitting prior to calling.
- **Interrupt management:** This function does not await any interrupts. It will be affected by interrupts.
- **Resources:** TPM, SIMOPT2, RFM
- **Input parameters:**
  - None
- **Returns:**
  - UINT8 u8WDIV: WDIV compensated value, or 80 h if the XTAL was not found.

Note: *This routine writes to SIMOPT2. Any configuration involving this register must be performed before calling this routine. Prior to calling this routine, the RFM must be turned on and the IPMS_PRECHARGE_EN function called. The execution of this routine changes the contents of RFM registers. Specifically note that RF direct mode will be selected after its execution and the TIMEOUT[1:0] bits in the RFPRECHARGE register set to value 2.*

3.2.17 IPMS_LFOCAL_BUSCLK

- **Description:** This function returns the calculated WDIV value for 1 sec base time PWU period. It uses the internal clock instead of the external RFM oscillator to measure the duration of one LFO cycle.
- **Stack size:** 11 bytes
- **Approximate duration:** 1919 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts.
- **Resources:** None.
- **Input parameters:**
  - None
- **Returns:**
  - Calculated WDIV value

3.2.18 UINT8 IPMS_MFOCAL (void)

- **Description:** Performs MFO cross-check verification. This function measures the bus clock relative to Dx, but first executes a test to verify the presence of the external XTAL. When the error is zero, it returns 128. Any deviation from this value should be considered an error. This result can then be used to estimate the error in the RFBT setting. The IPMS_MFOCAL subroutine cannot be used while the RFM is transmitting or the TPM1 is being used for another task. This function accesses and writes data to the **SIMOPT2** register. Because some of the bits in this register are write-once-only, it should be configured prior to calling this routine.
- **Stack size:** 11 bytes
- **Approximate duration:** 1803 µs
- **Power management:** This function executes entirely in RUN mode. It requires the MCU to be configured for a 4 MHz bus clock, and the RFM to be enabled, but not transmitting prior to calling.
- **Interrupt management:** This function does not await any interrupts. It will be affected by interrupts.
- **Resources:** IPMS, SIMOPT2, RFM
- **Input parameters:**
  - None
- **Returns:**
  - UINT8 u8Error: 128 when no error is found. Each LSB away from this value is equal to a 0.78 % error. For example, if u8Error = 125, the bus clock has a -2.34 % error, or is running at 3.9064 MHz. 255 is reserved as an error code for when the external XTAL is not present.

Note: *This routine writes to SIMOPT2. Any configuration involving this register must be performed before calling this routine. Prior to calling this routine, the RFM must be turned on and the IPMS_PRECHARGE_EN function called. The execution of this routine changes the contents of the RFM registers. Specifically note that RF direct mode will be selected after its execution and the TIMEOUT[1:0] bits in the RFPRECHARGE register set to value 2.*

3.2.19 UINT16 IPMS_WAVG (UINT8 u8PNew, UINT16 u16POld, UINT8 u8PAvg)

- **Description:** This subroutine calculates a new weigh average value for a given new and old measurement readings by using Equation 1.

$$u16NewAverage = \left(\left(u16POld \times \left(u8Avg - 1\right) + u8PNew\right) / \left(u8Avg\right)\right) \quad (1)$$

- **Stack size:** 12 bytes
- **Approximate duration:** 40.52 µs (average of 2), 44.57 µs (average of 4), 49.5 µs (average of 8), 54.1 µs (average of 16), 58.5 µs (average of 32).
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** N/A
- **Input parameters:**
  - UINT8 u8Avg: Weight of the average. This value can be 2, 4, 8, 16, 32; any other value returns an incorrect response.
  - UINT16 u16Pold: Old average.
  - UINT8 u8PNew: New value to include in the average.
- **Returns:**
  - UINT16 u8NewAverage: The resulting weighed average of both the old average and the new value (See Equation 1).

Note: *This function is not correctly implemented. The function exists and the user may execute the function but the function will not return the expected data described above. The application must not use the returned, incorrect data and the user should avoid using this function.*

3.2.20 Void IPMS_RF_ENABLE (UINT8 u8Switch)

- **Description:** This function enables or disables the RF module in the NPM8 and transfers adequate PLL trim data to the module. It should be called prior to any other RF operation.
- **Stack size:** 4 bytes
- **Approximate duration:** 364 µs when turning on; 11.2 µs when turning off.
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will be affected by interrupts.
- **Resources:** SIMOPT1, RFM
- **Input parameters:**
  - UINT8 u8Switch: Enable (nonzero) or disable (zero) RFM.
- **Returns:**
  - Void

Note: *This routine writes to SIMOPT1. Any configuration involving this register must be performed before calling this routine.*

3.2.21 Void IPMS_RF_RESET (void)

- **Description:** This function sends a central reset to the RFM and reloads PLL trim values into the module. It requires the RFM to have been enabled previously.
- **Stack size:** 3 bytes
- **Approximate duration:** 221 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** RFM
- **Input parameters:**
  - None
- **Returns:**
  - Void

3.2.22 Void IPMS_RF_READ_DATA (UINT8 u8Size, UINT8 *u8RAMBuffer, UINT8 u8RFMBuffer)

- **Description:** This function reads several consecutive bytes from the dedicated RFM buffer registers and copies them to a given address in RAM. It assumes that BUFF0 is location "0". The data is transferred from the LSB bit of the RFM data registers to the LSB of the target memory address (standard data bit order). This function manages the RFM's buffer paged memory.
  - In case the required buffer address is out of bounds, the routine returns "0" for that location.
- **Stack size:** 9 bytes
- **Approximate duration:** 196 µs (for 8 bytes, switching pages included).
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** RFM
- **Input parameters:**
  - UINT8 u8Size: Number of bytes to read.
  - UINT8 *u8RamBuffer: Target memory location.
  - UINT8 u8RFMBuffer: Buffer register (0 to 31) to read.
- **Returns:**
  - Void

3.2.23 Void IPMS_RF_READ_DATA_REVERSE (UINT8 u8Size, UINT8 *u8RAMBuffer, UINT8 u8RFMBuffer)

- **Description:** This function reads several consecutive bytes from the dedicated RFM buffer registers and copies them to a given address in RAM. It assumes that BUFF0 is location "0". The data is transferred from the LSB bit of each byte of the RFM data registers to the MSB of each of the bytes of the target memory address (reversed data bit order). This function manages the RFM's buffer paged memory.
  - In case the required buffer address is out of bounds, the routine returns "0" for that location.
- **Stack size:** 10 bytes
- **Approximate duration:** 236 µs (for 8 bytes, switching pages included).
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** RFM
- **Input parameters:**
  - UINT8 u8Size: Number of bytes to read.
  - UINT8 *u8RamBuffer: Target memory location.

– UINT8 u8RFMBuffer: Buffer register (0 to 31) to read.
- **Returns:**
  – Void

3.2.24 Void IPMS_RF_WRITE_DATA (UINT8 u8Size, UINT8 *u8RAMBuffer, UINT8 u8RFMBuffer)

- **Description:** This function copies several consecutive bytes from RAM into the dedicated RFM Output Buffer. It assumes that BUFF0 is location 0. The data is transferred from the LSB bit of RAM to the LSB of the RFM data register (standard data bit order). This function manages the RFM's buffer paged-memory.
  – In case the destination buffer address is out of bounds, the register value will not be written.
- **Stack size:** 12 bytes
- **Approximate duration:** 151 µs (for 8 bytes, switching pages included).
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** RFM
- **Input parameters:**
  – UINT8 u8Size: Number of bytes to write.
  – UINT8 u8RAMBuffer: Source memory location.
  – UINT8 u8RFMBuffer: Starting buffer register (0 to 31) to write.
- **Returns:**
  – Void

3.2.25 Void IPMS_RF_WRITE_DATA_REVERSE (UINT8 u8Size, UINT8 *u8RAMBuffer, UINT8 u8RFMBuffer)

- **Description:** This function copies several consecutive bytes from RAM into the dedicated RFM output buffer. It assumes that BUFF0 is location 0. The data is transferred from the LSB bit of each byte in RAM to the MSB of each byte in the RFM data register (reversed data bit order). This function manages the RFM's buffer paged-memory.
  – In case the destination buffer address is out of bounds, the register value is not written.
- **Stack size:** 11 bytes
- **Approximate duration:** 204 µs (for 8 bytes, switching pages included).
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It is not affected by interrupts.
- **Resources:** RFM
- **Input parameters:**
  – UINT8 u8Size: Number of bytes to write.
  – UINT8 u8RAMBuffer: Source memory location.
  – UINT8 8uRFMBuffer: Starting buffer register (0 to 31) to write.
- **Returns:**
  – Void

3.2.26 Void IPMS_RF_CONFIG_DATA (UINT16 *u16RFParam)

- **Description:** This function configures the RFM for transmission. It does not configure interframe wait times, which must be configured manually.
- **Stack size:** 4 bytes
- **Approximate duration:** 32 µs
- **Power management:** This function executes entirely in RUN mode.

- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** RFM
- **Input parameters:**
  - UINT16* u16RFParam Format as described in Table 25.
- **Returns:**
  - Void

Table 25. u16RFParam array format

Index	Description
0	See Table 26 for description
1	PLLA value
2	PLLB value

Table 26. Description of element 0 in the u16RFParam array

Bits	Description
15:8	Prescaler value. Described in the data sheets as RFCR0.
7	End of message- If '1', EOM is set, if '0', it is not set.
6	Polarity bit - If '1', polarity is inverted, If '0', it is noninverted.
5:4	Not used.
3:2	Encoding value.
1	Frequency selection - If '1', RFM is configured for 434 MHz, if '0', it is configured for 315 MHz.
0	Modulation - If '1', RFM is configured for FSK, if '0' it is configured for OOK.

3.2.27 Void IPMS_RF_SET_TX (UINT8 u8BufferSize)

- **Description:** This function allows the RFM to transmit data previously loaded in the buffer. It should be called after the RF module has been enabled and configured.
- **Stack size:** 4 bytes
- **Approximate duration:** 13 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** RFM
- **Input parameters:**
  - UINT8 u8BufferSize: Number of bits in the buffer -1 (for example to transmit 1 bit, u8BufferSize should equal 0).
- **Returns:**
  - Void

3.2.28 Void IPMS_READ_ID (UINT8 *u8Code)

- **Description:** Copies the UniqueID of the device and firmware version stored in firmware flash to RAM.
- **Stack size:** 2 bytes
- **Approximate Duration:** 17 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.

- **Resources:** N/A
- **Input parameters:**
  - UINT8 *u8Code: RAM location where data is copied. Table 27 describes the format of the 6 bytes returned.

Table 27. u8Code format

Index	Description
0	Library version
1	Derivative descriptor
2 to 5	32-bit UniqueID

- **Returns:**
  - Void

3.2.29 Void IPMS_RF_DYNAMIC_POWER (UINT8 u8CompT, UINT8 u8CompV, UINT8* pu8PowerManagement)

- **Description:** Depending on the passed parameters, this function can:
  - Force the RF power setting (RFCR2_PWR) to a passed value (when BIT5 of u8PowerManagement is clear).
  - Set the RF power setting (RFCR2_PWR) dynamically based on voltage, temperature, and current carrier frequency (when BIT5 of u8PowerManagement is set). The target output level is 3 dBm across all voltages and temperatures, with some small variations. When this option is engaged, the routine limits settings to valid PWR settings - if the resulting value is above the maximum allowed setting, the setting is set to maximum; if the resulting value is less than the minimum allowed setting, the setting is set to the minimum.
  - When BIT5 of u8PowerManagement is set, find the best RF power setting (RFCFR2_PWR) dynamically based on voltage, temperature, and current carrier frequency to target 3 dBm as actual output power. This value of 3 dBm can be increased or decreased in given temperature ranges using the offsets (0.5 dBm/count) in the pu8PowerManagement array.
  - Similar to the case above, the user can specify a target power region with an offset.
- **Stack size:** 21 bytes
- **Approximate duration:** 140 µs when using voltage, temperature; 22 µs when the power step is passed.
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** RFM
- **Input parameters:**
  - UINT8 u8CompT: Compensated temperature reading.
  - UINT8 u8CompV: Compensated voltage reading.
  - UINT8* pu8PowerManagement: This is a pointer to an array as described in Table 28 and Table 29:
- **Returns:**
  - Void

Note: *The RF module must be turned on prior to calling this routine.*

Table 28. *pu8PowerManagement format

Index value	Description
0	Dynamic compensation switch as described in Table 29.
1	Offset step for power target when the temperature is higher than 92 °C. Negative values are admitted.
2	Offset step for power target when the temperature is lower than 92 °C and higher than 60 °C. Negative values are admitted.

Table 28. *pu8PowerManagement format*...*continued*

Index value	Description
3	Offset step for power target when the temperature is lower than 60 °C and higher than 43 °C. Negative values are admitted.
4	Offset step for power target when the temperature is lower than 43 °C and higher than 25 °C. Negative values are admitted.
5	Offset step for power target when the temperature is lower than 25 °C and higher than 0 °C. Negative values are admitted.
6	Offset step for power target when the temperature is lower than 0 °C and higher than -20 °C. Negative values are admitted.
7	Offset step for power target when the temperature is lower than -20 °C. Negative values are admitted.

Table 29. pu8PowerManagement format

Bit	Description
MSB	Not used.
BIT6	Not used.
BIT5	Dynamic compensation enable. If set, the function decides what the optimal power setting is based on voltage and temperature; In this case, values stored in the pu8PowerManagement array, and corresponding to the temperature range will be added to the found target. If clear, BIT4:0 will be used to set the power level directly.
BIT4:0	When BIT5 is clear, the value passed here will be used to set the RF power step in the RFCR2 register directly.

3.2.30 Void IPMS_MSG_INIT (void)

- **Description:** This function is to be called before using any MSG routine. It initializes PTA1 and PTA0 to their correct initial state for a simulated SPI.
- **Stack size:** 2 bytes
- **Approximate duration:** 9 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** Pins PTA1 and PTA0
- **Input parameters:**
  - None
- **Returns:**
  - Void

3.2.31 UINT8 IPMS_MSG_WRITE (UINT8 u8SendByte)

- **Description:** This function is in charge to write a message at a network level via an emulated serial interface on PTA1 and PTA0. As the central, the NPM8 manages the clock on PTA1. On the rising edge of the clock, the module puts down a new data bit on PTA0 (programmed as output), MSB first.
- **Stack size:** 2 bytes
- **Approximate duration:** 78 µs
- **Power management:** This function executes entirely in RUN mode.

- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** Pins PTA1 and PTA0
- **Input parameters:**
  - UINT8 u8SendByte: Byte to be outputted through the emulated serial interface
- **Returns:**
  - UINT8 u8ReadByte: Incoming byte from the emulated serial interface

3.2.32 UINT8 IPMS_MSG_READ (void)

- **Description:** This function is in charge to read any incoming message at a network level via an emulated serial interface on PTA1 and PTA0. As the central, the NPM8 manages the clock on PTA1. On falling edge of the clock, the module reads a new data bit on PTA0 (programmed as input), MSB first.
- **Stack size:** 2 bytes
- **Approximate duration:** 78 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** Pins PTA1 and PTA0.
- **Input parameters:**
  - None
- **Returns:**
  - UINT8 u8ReadByte: Incoming byte from the emulated serial interface

3.2.33 UINT8 IPMS_CHECKSUM_XOR (UINT8 *u8Buffer, UINT8 u8Size, UINT8 u8Checksum)

- **Description:** Calculates a checksum for the given buffer based on XOR operations.
- **Stack size:** 5 bytes
- **Approximate duration:** 78 µs for 8 bytes of data.
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** N/A
- **Input parameters:**
  - UINT8 *u8Buffer: Buffer where data is located.
  - UINT8 u8Size: Size of buffer (in bytes).
  - UINT8 u8Checksum: Previous checksum. This argument is useful when the function is used recursively. It must equal "0" if there is no previous data.
- **Returns:**
  - UINT8 u8NewChecksum: New calculated checksum.

3.2.34 UINT8 IPMS_CRC8 (UINT8 *u8Buffer, UINT8 u8Poly, UINT8 u8MBitSize, UINT8 u8Remainder)

- **Description:** Calculates a CRC8 on a portion of the designated area.
- **Stack size:** 12 bytes
- **Approximate duration:** 780 µs for 8 bytes (64 bits) of data.
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** The function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** N/A
- **Input parameters:**
  - UINT8 *u8Buffer: Buffer where data is located.

– UINT8 u8Poly: Polynomial to be is used for calculating the CRC8.
– UINT8 u8MBitSize: Size of the designated buffer (in bits).
– UINT8 u8Remainder: Initial remainder. This argument is useful when the function is used recursively. It must equal "0" if there is no previous data.

- **Returns:**
  – UINT8 u8NewCRC: New calculated CRC8.

3.2.35 UINT16 IPMS_SQUARE_ROOT (UINT16 u16Process)

- **Description:** Calculates a two-digit remainder of (square root * 10) using a fast algorithm.
- **Stack size:** 49 bytes
- **Approximate duration:** 362 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** N/A
- **Input parameters:**
  – UINT16 u16Process: The number from which to get the square root from.
- **Returns:**
  – UINT16 Root of the number * 10.

Note: *The result may include an error due to truncation in the calculation.*

3.2.36 Void IPMS_LF_ENABLE (UINT8 u8Switch)

- **Description:** Enables/disables the LFR module; Loads best-case-known LF settings for NXP-only LF registers by calling the vnfLFRConfigFactoryRegs() function described in Section 2.5.
- **Stack size:** 5 bytes
- **Approximate duration:** 30 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will not be affected by interrupts.
- **Resources:** LFR
- **Input parameters:**
  – UINT8 u8Switch: Enable (nonzero) or disable (zero) LFR.
- **Returns:**
  – void

3.2.37 UINT8 IPMS_LF_READ_DATA (UINT8 *u8Buffer, UINT8 u8Count)

- **Description:** Once the user has configured and enabled the LFR, it is customary to go into a low-power state mode and wait for a datagram. After the first byte of an incoming datagram is successfully received, this function should be called immediately. The function receives the complete datagram and place it in RAM. Be careful to call the function upon reception of the first data byte (LFDRF flag) and not upon detection of the ID (LFIDF flag) in case the LFIDIE is enabled. This function assumes that the LFR module is configured accordingly for a Manchester reception, that the module's interrupts are enabled, and that the first byte has already been received and is waiting in the LFR received buffer. While waiting for the next byte, this function goes into STOP4. If the byte, for an unexpected reason, is not received, this function has the following built-in timeout: After five continuous non-LFR interrupts, the function will assume a failed LFR reception and exit. To leave the routine as soon as possible after reception of all the data bytes, NXP recommends enabling the LF error interrupt (LFERIE). In summary, the two necessary interrupts to be enabled are LFDRIE and LFERIE.
- **Description:**
- **Stack size:** 7 bytes

- **Approximate duration:** Data dependant; ~2 ms per byte received.
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the LFR interrupt to wake up from STOP mode.
- **Resources:** LFR
- **Input parameters:**
  - UINT8 *u8Buffer: RAM Buffer where data is placed.
  - UINT8 u8Count: Number of bytes expected.
- **Returns:**
  - UINT8 u8BytesReceived: Actual number of bytes received.

Note: *This function requires ~24 µs from the moment that it is called to the moment the first byte is copied into the RAM buffer. The user must consider this time when designing their firmware.*

Note: *This function should be called only after the first byte of the incoming datagram has been received. If it is called before receiving the first byte, the value returned by the function is unpredictable.*

3.2.38 UINT8 IPMS_WIRE_AND_ADC_CHECK (UINT8 u8TestMask)

- **Description:** This function checks if there is any bonding wire failure between the embedded core and the P-cell; or between the core and the g-cell. It will also perform an ADC test, which consists of taking two reference measurements, ground and V_{DD}, using internal channels and comparing them with the expected results. It can only be called when the device is in parking or static mode. When configuring for a P-cell or g-cell wire check, interrupts must be enabled before calling this routine. In no issues found, 0 will be returned, else it sets status flags as follows:
  - On P-cell wire-bond error, sets a pressure error flag.
  - On g-cell wire-bond error, sets an acceleration error flag.
  - On ADC error, sets the ADCERR flag.
- **Stack size:** Up to 37 bytes
- **Approximate duration:** 9815 µs (all checks), 121 µs (ADC only), 3,296 µs (P-cell only), 3227 µs (X-cell or Z-cell only).
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function utilizes the ADC interrupt to wake up from STOP mode.
- **Resources:** ADC, SMI (for g-cell, P-cell checks), internal bond wires
- **Input parameters:**
  - UINT8 u8TestMask: This variable determines what checks are performed as described by Table 30.

Table 30. u8TestMask format

u8TestMask Bit	Description
1	Reserved
2	If set, a P-cell wire-bond check is performed
3	If set, g- Xcell wire-bond check is performed
4	If set, g-Zcell wire-bond check is performed
5 to 6	Reserved
7	If set, the ADC checks performed

Note: *Users should not set bits 0, 1, 5, or 6. The results returned, according to Table 31, may be ambiguous, including indicating false passing or false failing conditions.*

- **Returns:**
  - UINT8 u8Status: Status flags as described in Table 31.

Table 31. u8Status valid values for IPMS_WIRE_AND_ADC_CHECK

u8TestMask Bit	Description
0	Always clear
1	Always clear
2	If set, P-cell wire-bond error detected
3	if set, g-xcell error is returned
4	if set, g-zcell error is returned
5 to 6	Always clear
7	If set, ADC error detected

3.2.39 Void IPMS_FLASH_WRITE (UINT16 u16Address, UINT8* u8Buffer, UINT8 u8Size)

- **Description:** This function writes consecutive bytes from a given address in memory to a specified location in FLASH. Addresses $FD40 - $FDFF are not valid targets in this function
- **Stack size:** 15 bytes
- **Approximate duration:** 1270 µs for 8 bytes of data.
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It will be affected by interrupts.
- **Resources:** 59 bytes of global RAM locations
- **Input parameters:**
  - UINT16 u16Address: Flash starting address
  - UINT8 *u8Buffer: Source memory address
  - UINT8 u8Size: Number of data bytes to be written. Must be strictly greater than zero.
- **Returns:**
  - void

Note: *This routine overwrites the contents of 59 bytes of RAM allocated to the FLASHING_ALGO_LOCATION_IN_RAM section. In NXP demo projects, this section is mapped to addresses 0090h to 00CAh.*

3.2.40 UINT8 IPMS_FLASH_ERASE (UINT16 u16Address)

- **Description:** This function erases 1 page (512 bytes) of flash at a time. Addresses $FC00 - $FDFF are not valid targets in this function.
- **Stack size:** 11 bytes
- **Approximate duration:** 22,750 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It may be affected by interrupts.
- **Resources:** 59 bytes of global RAM locations.
- **Input parameters:**
  - UINT16 u16Address: any given address. The whole page where this address resides will be erased (for example, if u16Address = D234h, the contents of addresses D200h - D3FFh will be erased).
- **Returns:**
  - Zero if the page was erased successfully; else, one.

Note: *This routine overwrites the contents of 59 bytes of RAM allocated to the FLASHING_ALGO_LOCATION_IN_RAM section. In NXP demo projects, this section is mapped to addresses 0090h to 00CAh.*

3.2.41 void IPMS_MULT_SIGN_INT16 (INT16 i16Mult1, INT16 i16Mult2, INT32* pi32Result)

- **Description:** This function will multiply two signed 16-bit numbers together.
- **Stack size:** 17 bytes
- **Approximate duration:** 68.1 µs
- **Power management:** This function executes entirely in RUN mode.
- **Interrupt management:** This function does not await any interrupts. It should not be affected by interrupts.
- **Resources:** N/A
- **Input parameters:**
  - INT16 i16Mult1: First multiplier
  - INT16 i16Mult2: Second multiplier
  - INT32* pi32Result: Pointer to a 32-bit variable where the result will be stored.
- **Returns:**
  - Void.

3.2.42 UINT8 IPMS_VREG_CHECK (UINT8 u8WaitTime, UINT16 u16LimitDelta)

- **Description:** This function verifies if the part has a capacitor properly connected on the VReg pin. This is done by starting an RF transmission in Buffer mode, once the transmission is done, taking a first ADC reading of the VReg pin, awaiting a preestablished amount of time, then taking a second ADC reading of the VReg pin. The difference between the two readings is then calculated. If it stays below a certain limit, it means that the capacitor is properly connected to the VReg pin.
- **Stack size:** 30 bytes
- **Approximate duration:** 28,500 µs using default values; time varies depending on user input.
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses an ADC interrupt to wake up from STOP mode.
- **Resources:** TPM, RFM, SPMSC2.
- **Input parameters:**
  - UINT8 u8WaitTime: Amount of time to wait between the two ADC readings (in ms). If zero, it is assumed that a 470 nF capacitor is being used and the default wait time of 25 ms for this capacitor is used.
  - UINT8 u8LimitDelta: This value (in ADC counts) will determine whether the V_{REG} pin passes the test or it does not. It depends on the capacitor value and on the value of u8WaitTime, and must be obtained through characterization. If zero, it is assumed that a 470 nF capacitor is being used and the default limit of $50 is used.
- **Returns:**
  - UINT8 u8Status: If clear, the function has detected good contact with the capacitor; if one, the capacitor has failed the test.

Note: *Write-once register SPMSC2 is used. Also note that calling this function starts an RF transmission for ~3 ms. Previously set RF settings, such as carrier frequency and PLL dividers, are respected in this short RF burst. Before exiting this function, the RF module is shut down.*

3.2.43 UINT8 IPMS_E_READ_ACCEL_X (UINT16* pu16UUMA, UINT8 u8Avg, UINT8 u8Filter, UINT8 u8OffsetIndex, UINT8 u8delay)

- **Description:** This function is an extension of IPMS_READ_ACCEL_X which takes sampling delay as application options.
- **Stack size:** 37 bytes
- **Approximate duration:**

Table 32. Execution time for IPMS_E_READ_ACCEL_X

Initial delay	2048 µs	1024 µs	512 µs
Execution time for u8Avg = 1	2.37 ms	1.35 ms	0.852 ms

u8Avg	1	2	3
Execution time for initial delay = 2048 µs and Subsequent delay = 128 µs	2.37 ms	2.54 ms	2.86 ms

- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from Stop mode.
- **Resources:** SMI, ADC, Internal bond wires.
- **Input parameters:**
  - UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3. Only the 12-bit uncompensated acceleration result is updated.
  - UINT8 u8Avg: The number of measurements to average into one result. The value can be set to 1, 2, 4, 8, or 16.
  - UINT8 u8Filter: Sets up the acceleration measurement based on Table 17.
  - UINT8 u8OffsetIndex: Selects the offset setting in normal dynamic offset mode. The default index is 7. The valid range is 0 to 15.
  - UINT8 u8Delay: composition of initial and subsequent delay, for example, SAMP_DLY_SUBS_512US| SAMP_DLY_INIT_2048US. See Table 7 and Table 8.
- **Returns:** UINT8 u8Status: Valid error flags/outputs are described in Table 18.

3.2.44 UINT8 IPMS_E_READ_PRESSURE (UINT16* pu16UUMA, UINT8 u8Avg, UINT8 u8Filter, UINT8 u8Delay)

- **Description:** This function is an extension of IPMS_READ_PRESSURE which takes the sampling delay as an application option.
- **Stack size:** 36 bytes
- **Approximate duration:** 219 µs

Table 33. Execution time for IPMS_E_READ_PRESSURE

Initial delay	2048 µs	1024 µs	512 µs
Execution time for u8Avg = 1	2.37 ms	1.35 ms	0.852 ms

u8Avg	1	2	3
Execution time for initial delay = 2048 µs and Subsequent delay = 128 µs	2.37 ms	2.54 ms	2.86 ms

- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** SMI, ADC, Internal bond wires
- **Input parameters:**
  - UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3. Only the 12-bit uncompensated acceleration result is updated.
  - UINT8 u8Avg: The number of measurements to average into one result. The value can be set to 1, 2, 4, 8, or 16.
  - UINT8 u8Filter: Filter setting based on Table 17.
  - UINT8 u8Delay: composition of initial and subsequent delay, for example, SAMP_DLY_SUBS_512US| SAMP_DLY_INIT_2048US. See Table 7 and Table 8.
- **Returns:** UINT8 u8Status: Valid error flags/outputs are described in Table 14.

3.2.45 UINT8 IPMS_E_READ_ACCEL_Z (UINT16 pu16UUMA, UINT8 u8Avg, UINT8 u8Filter, UINT8 u8OffsetIndex, UINT8 delay)

- **Description:** This function is an extension of IPMS_READ_ACCEL_Z which takes sampling delay as application options.
- **Stack size:** 37 bytes
- **Approximate duration:**

Table 34. Execution time for IPMS_E_READ_ACCEL_Z

Initial delay	2048 µs	1024 µs	512 µs
Execution time for u8Avg = 1	2.37 ms	1.35 ms	0.852 ms

u8Avg	1	2	3
Execution time for initial delay = 2048 µs and Subsequent delay = 128 µs	2.37 ms	2.54 ms	2.86 ms

- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** SMI, ADC, Internal bond wires
- **Input parameters:**
  - UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3. Only the 12-bit uncompensated acceleration result is updated.
  - UINT8 u8Avg: The number of measurements to average into one result. The value can be set to 1, 2, 4, 8, or 16.
  - UINT8 u8Filter: Sets up the acceleration measurement based on Table 17.
  - UINT8 u8OffsetIndex: Selects the offset setting in normal dynamic offset mode. The default index is 7. The valid range is 0 to 15.
  - UINT8 u8Delay: composition of initial and subsequent delay, for example, SAMP_DLY_SUBS_512US| SAMP_DLY_INIT_2048US. See Table 7 and Table 8.
- **Returns:** UINT8 u8Status: Valid error flags/outputs are described in Table 14.

3.2.46 Void IPMS_E_FRC_ENABLE (UINT8 u8ClrRes)

- **Description:** This function enables Free Running Counter.
  - **Stack size:** 3 bytes

– **Approximate Duration: 1.3** ms
– **Power management:** This function executes entirely in RUN.
– **Interrupt management:** None
– **Resources: FRC**
– **Input parameters:**
  – UINT8 u8ClrRes – clear or not the FRC counter
  – 0 – clear the FRC counter,
  – not 0 – FRC counter is not cleared.
– **Returns:** Void

3.2.47 Void IPMS_E_FRC_DISABLE (void)

- **Description:** This function disables the Free running counter.
- **Stack size:** 0 bytes
- **Approximate duration:** 1 ms
- **Power management:** This function executes entirely in RUN
- **Interrupt management:** None
- **Resources:** FRC
- **Input parameters:** Void
- **Returns:** Void

3.2.48 Void IPMS_E_FRC_CLEAR (void)

- **Description:** This function clears the FRC counter register
- **Stack size:** 0 bytes
- **Approximate duration:** 9 µs
- **Power management:** This function executes entirely in RUN
- **Interrupt management:** None
- **Resources:** FRC
- **Input parameters:** Void
- **Returns:** Void

3.2.49 Void IPMS_E_FRC_READ (UINT16 *pu16Count)

- **Description:** This function reads the FRC counter register
- **Stack size:** 2 bytes
- **Approximate duration:** 10 µs
- **Power management:** This function executes entirely in RUN.
- **Interrupt management:** None
- **Resources:** FRC
- **Input parameters:**
  – UINT16 *pu16Count - pointer to memory location to save the FRC count
- **Returns:** Void

3.2.50 UINT8 IPMS_E_FRC_CALIB (UINT16 *pu16usPerPrd)

- **Description:**
  – This function calculates the number of microseconds per LFO period using the 500kHz signal coming from the external XTAL as reference. On exit, the FRC remains enabled. The function takes care of enabling and

disabling the RF block, but does not perform a precharge. Users should call the precharge function before calling the IPMS_FRC_CALIB function.

- **Stack size:** 23 bytes
- **Approximate duration:** 9.1 µs
- **Power management:** This function executes entirely in RUN
- **Interrupt management:** None
- **Resources:** This function uses the FRC block, the timer module TPM1, and the 500 kHz reference clock generated by the RF module. On entry, it expects that the TPM1 module is in reset state in free-running timer counter mode with modulo counting disabled. Before exiting, the function disables the TPM1 and RF module, and keeps the FRC enabled.
- **Input parameters:** UINT16 *pu16usPerPrd - pointer to memory location to save number of microseconds per LFO period.
- **Returns:** UINT8 u8Status - status/error flag:
  - 0 valid value returned in *pu16usPerPrd
  - 0x80 indicates an XTAL error.
  - Other none 0 – timeout, contents of *pu16usPerPrd is not valid.

Note: *This routine writes to SIMOPT2. Any configuration involving this register must be performed before calling this routine. The execution of this routine changes the contents of the RFM registers. Specifically note that RF Direct Mode will be selected after its execution and the TIMEOUT[1:0] bits in the RFPRECHARGE register set to value 2.*

3.2.51 UINT8 IPMS_E_FRC_CALIB_BUSCLK (UINT16 *pu16usPerPrd)

- **Description:** This function calculates the number of microseconds per LFO period using the Bus clock as a reference. On exit FRC remains enabled.
- **Stack size:** 18 bytes
- **Approximate duration:** 5.3 ms
- **Power management:** This function executes entirely in RUN.
- **Interrupt management:** None
- **Resources:** This function uses the FRC block, the timer module TPM1, and the BUSCLK.
- **Input parameters:** UINT16 *pu16usPerPrd - pointer to memory location to save number of microseconds per LFO period.
- **Returns:** UINT8 - status/error flag:
  - 0 if the operation is successful,
  - Non-0, indicating the function exited on timeout.

3.2.52 UINT8 IPMS_PRECHARGE_EN (void)

- **Description:** Users should call the IPMS_PRECHARGE_EN function every time the RF block is used to reduce the current peak due to the start of the analog regulator. Specifically, call the function before the 500 kHz Dx signal is generated and before starting an RF transmission. This function manages the precharge feature and assures the precharge and completes with an indicated status. It should be called in place of or in addition to setting the AREGPC bit. The precharge cannot be performed by setting AREGPC bit only; this function must be called. The RF block must be enabled prior to calling this function.
- **Stack size:** 3 bytes
- **Approximate duration:** 619 µs, the time varies depending on the number of times the retry happens
- **Power management:** This function executes entirely in RUN
- **Interrupt management:** None
- **Resources:** RF Block

- **Input parameters:** None
- **Returns:** UINT8 - status/error flag
  - 0 for Success
  - 1 for the precharge does not complete and exceed the amount of retry.

Note: *The RF registers are reset to their default values inside the function.*

3.2.53 Void IPMS_E_ACTIVATE_CHANNEL (UINT8 u8Filter, UINT8 u8OffsetIndex, UINT8 u8Delay, channel_t eChannel)

- **Description:** This function activates the channel based on the channel selected. It enables the 12-bit ADC with proper configuration, sets the appropriate trim information into the trim register, and enables the LPDM and acquisition.
- **Stack size:** 18 bytes
- **Approximate duration:** 140 µs
- **Power management:** This function executes entirely in RUN
- **Interrupt management:** None
- **Resources:** SMI and ADC
- **Input parameters:**
  - UINT8 u8Filter: Sets up the acceleration or pressure measurement based on Table 17.
  - UINT8 u8OffsetIndex: Selects the offset setting in normal dynamic offset mode. The default index is 7 and the valid range is 0 to 15. This parameter is not used when the pressure channel is selected.
  - UINT8 u8Delay: composition of initial and subsequent delay, for example, SAMP_DLY_SUBS_512US| SAMP_DLY_INIT_2048US. See Table 7 and Table 8.
  - Channel_t eChannel: Channel selection, for selecting pressure channel, pass the CHANNEL_PRESSURE enum, see Table 35
- **Table 35. eChannel values**

eChannel value	Numerical value
CHANNEL_PRESSURE	0
CHANNEL_Z	1
CHANNEL_X	2

- **Returns:** void

Note: *Putting the device into STOP4 mode after activating the particular channel is the responsibility of the application code.*

3.2.54 Void IPMS_E_DEACTIVATE_CHANNEL (void)

- **Description:** This function deactivates the currently active channel
- **Stack size:** 8 bytes
- **Approximate duration:** 10 µs
- **Power management:** This function executes entirely in RUN.
- **Interrupt management:** None
- **Resources:** SMI and ADC
- **Input parameters:** Void
- **Returns:** Void

3.2.55 UINT8 IPMS_E_READ_CONT_ACCEL_X (UINT16* u16UUMA)

- **Description:** This function is to be called after the x-axis channel has been activated. It checks if an ADC interrupt occurred, meaning a new x-axis measurement is available. If so, the measurement value is stored in the UUMA array passed as a parameter and the function returns 0 otherwise it returns 1 for an error.
- **Stack size:** 8 bytes
- **Approximate duration:** 18 µs
- **Power management:** This function executes entirely in RUN
- **Interrupt management:** None
- **Resources:** SMI and ADC
- **Input parameters:**
  - • UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3. Only the 12-bit uncompensated acceleration result is updated.
- **Returns:** UINT8 0 for success and 1 for an error.

Note: *Data is ideally to be read under STOP4. Putting the device into STOP4 mode after activating the X channel is the responsibility of the application code.*

3.2.56 UINT8 IPMS_E_READ_CONT_ACCEL_Z (UINT16* u16UUMA)

- **Description:** This function is to be called after the z-axis channel has been activated. It checks if an ADC interrupt occurred, meaning a new z-axis measurement is available. If so, the measurement value is stored in the UUMA array passed as a parameter and the function returns 0 otherwise it returns 1 for an error
- **Stack size:** 8 bytes
- **Approximate duration:** 18 µs
- **Power management:** This function executes entirely in RUN.
- **Interrupt management:** None
- **Resources:** SMI and ADC
- **Input parameters:**
  - UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3. Only the 12-bit uncompensated acceleration result is updated.
- **Returns:** UINT8 0 for success and 1 for an error

Note: *Data is ideally to be read under STOP4. Putting the device into STOP4 mode after activating the Z channel is the responsibility of the application code.*

3.2.57 UINT8 IPMS_E_READ_CONT_PRESSURE (UINT16* u16UUMA)

- **Description:** This function is to be called after the pressure channel has been activated. It checks if an ADC interrupt occurred, meaning a new pressure measurement is available. If so, the measurement value is stored in the UUMA array passed as a parameter and the function returns 0 otherwise it returns 1 for an error.
- **Stack size:** 8 bytes
- **Approximate duration:** 18 µs
- **Power management:** This function executes entirely in RUN.
- **Interrupt management:** None
- **Resources:** SMI and ADC
- **Input parameters:**
  - • UINT16 *u16UUMA: Pointer to UUMA (as described in Section 2.3. Only the 12-bit uncompensated acceleration result is updated.
- **Returns:** UINT8 0 for success and 1 for an error

Note: *Data is ideally to be read under STOP4. Putting the device into STOP4 mode after activating the pressure channel is the responsibility of the application code.*

3.2.58 UINT8 IPMS_E_READ_DYNAMIC_ACCEL_Z (UINT16* pu16UUMA, UINT8 u8Filter, UINT8* pu8Offset, UINT8 u8Delay)

- **Description:** This function automatically executes a IPMS_E_READ_ACCEL_Z measurement with a given initial dynamic offset. If the result is too high or too low, it changes the dynamic offset value and reexecute IPMS_E_READ_ACCEL_Z until a) the result is valid or b) the result is railed high or low and there are no more offset steps. Offset and uncompensated acceleration inside the UUMA are updated.
- **Stack size:** 57 bytes
- **Approximate duration:** 2100 µs when the starting offset is in target with an initial delay of 2048 µs and subsequent delay of 512 µs; 21,000 µs when the offset is 10 steps away.
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from Stop mode.
- **Resources:** SMI, ADC, internal bond wires.
- **Input parameters:**
  - UINT16* pu16UUMA, Pointer to the UUMA. Uncompensated acceleration is updated accordingly.
  - UINT8 u8Filter: Sets up the acceleration measurement based on Table 17.
  - UINT8* pu8Offset: Pointer to initial offset step to load. Valid offset steps range from 0 to 15 and are described in the devices data sheet. An updated offset value is returned at the end of the function. In the case the acceleration is too high or too low and the function has run out of offset steps, a value of 255 (“0 to 1“) or 16 (”15 + 1”) shall be returned.
  - UINT8 u8Delay: composition of initial and subsequent delay, for example, SAMP_DLY_SUBS_512US| SAMP_DLY_INIT_2048US. See Table 7 and Table 8.
- **Returns:** UINT8 returns 0 for success and sees Section 3.2.11 for more information on the format of this status byte.

3.2.59 UINT8 IPMS_E_READ_DYNAMIC_ACCEL_X (UINT16* pu16UUMA, UINT8 u8Filter, UINT8* pu8Offset, UINT8 u8Delay)

- **Description:** This function automatically executes a IPMS_E_READ_ACCEL_X measurement with a given initial dynamic offset. If the result is too high or too low, it changes the dynamic offset value and reexecute IPMS_E_READ_ACCEL_X until a) the result is valid or b) the result is railed high or low and there are no more offset steps. Offset and uncompensated acceleration inside the UUMA are updated.
- **Stack size:** 57 bytes
- **Approximate duration:** 2100 µs when the starting offset is in target with an initial delay of 2048 µs and subsequent delay of 512 µs; 21,000 µs when the offset is 10 steps away.
- **Power management:** This function requires the core to be configured for STOP4 mode and running at full bus speed.
- **Interrupt management:** This function uses the ADC interrupt to wake up from STOP mode.
- **Resources:** SMI, ADC, internal bond wires.
- **Input parameters:**
  - UINT16* pu16UUMA, Pointer to the UUMA. Uncompensated acceleration is updated accordingly.
  - UINT8 u8Filter: Sets up the acceleration measurement based on Table 17.
  - UINT8* pu8Offset: Pointer to initial offset step to load. Valid offset steps range from 0 to 15 and are described in the devices data sheet. An updated offset value is returned at the end of the function. In the case the acceleration is too high or too low and the function has run out of offset steps, a value of 255 (“0 to 1“) or 16 (”15 + 1”) shall be returned.

  - UINT8 u8Delay: composition of initial and subsequent delay, for example, SAMP_DLY_SUBS_512US| SAMP_DLY_INIT_2048US. See Table 7 and Table 8.
- **Returns:** UINT8 See Section 3.2.8, IPMS_READ_ACCEL_X for more information on the format of this status byte.

4 Revision history

Table 36. Revision history

Document ID	Release date	Description
UM12293.v.1	25 July 2025	Initial version

Legal information

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.

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.

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

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 https://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.

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.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

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.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

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

Figures

Contents