AN12524

Migration from FXTH87/87E to NTM88

Rev. 2 — 10 December 2019 Application note

1 Introduction

This document presents the main differences between FXTH87/87E and NTM88 from a software application point of view.

Note: *Unless otherwise stated, the term "NTM88" refers to both the engineering samples (PTM88) and the production samples (NTM88). In* Section 4 "Memory map and application model"*, a distinction is made between the engineering and production samples. In all other sections, the information applies to both the engineering and production samples.*

2 Getting started

The IDE to develop NTM88 applications remains CodeWarrior. A patch[1] must be installed to have access to the NTM88 target.

From a software application point of view, NTM88 supports only Library Model Applications. NXP does not program NTM88 samples with embedded firmware functions. Users familiar with Firmware Model Applications can refer to AN12523, *Firmware versus Library model applications*. AN12523 provides information on the differences between these two models and how to migrate applications from a firmware model to a library model.

3 Pressure and acceleration transfer functions

FXTH87 and FXTH87E have different transfer functions for pressure and acceleration. For pressure and acceleration, FXTH87 and FXTH87E use a 10-bit representation for raw measurement of pressure and acceleration. However, for compensated measurement, a 9-bit representation is used.

NTM88 uses a 12-bit representation for raw measurement and a 10-bit representation for compensated measurement of pressure and acceleration.

Refer to the appropriate data sheet for actual compensated transfer functions because they differ from one part number to another.

1 Refer to *Installing NTM88 patch CW11* in the documentation provided with the NTM88 demo projects for information on patch installation and CodeWarrior versions compatible with this patch.

Table 1. Bit representation of raw pressure and acceleration and compensated pressure and acceleration

	Representation of raw pressure and acceleration	Representation of compensated pressure and acceleration
FXTH87 and FXTH87E	10 bits	9 bits
NTM88	12 bits	10 bits

Note: *The single axis derivative of the FXTH87/87E is a single Z-axis whereas with NTM88, the single axis derivative is a single X-axis.*

4 Memory map and application model

The memory map is the same between the FXTH87 and FXTH87E, but different for the NTM88.

During the production of FXTH87 and FXTH87E devices, NXP programs the firmware and trim between E000h and FFFFh. The user may choose to develop either firmware-based applications or library-based applications. For firmware-based applications, the embedded firmware is used by the application. With library-based applications, the user erases the embedded firmware and reprograms the functions used by the application.

The PTM88xxxS alpha-engineering samples have been shipped with an embedded firmware. This is indicated for reference only as these samples are not distributed anymore. NXP programs PTM88xxx5 and NTM88 production samples with trim only. Without programmed firmware functions, only the library model is applicable for PTM88xx5 and NTM88.

Table 2. Engineering and production sample firmware and programming

	Engineering samples	Production samples
FXTH87 and FXTH87E	N/A *(already released to production)*	Firmware and trim programmed by NXP between E000h and FFFFh *Firmware and library model can be used.*
PTM88xxxS (alpha samples – not provided anymore)	Firmware and trim programmed by NXP between E800h and FFFFh.	N/A (alpha samples not released to production)
PTM88xxx5 and NTM88	No firmware function programmed. NXP programs only trim between FD40h and FDFFh. See Section 4.1. *Only library model can be used.*	

4.1 Overview of the memory maps

FXTH87 and FXTH87E samples shipped by NXP are programmed with an embedded firmware containing:

- the firmware functions,
- a jump table allowing user application calls to the firmware functions,
- trim coefficients and
- firmware interrupt vectors.

For these products, customers may choose to

- use the firmware model by keeping the embedded firmware

- use the library model by erasing the embedded firmware (trim page excluded) and adding firmware functions used by the application from the firmware library

Refer to AN12523, *Firmware versus Library model applications* for additional information.

In contrast, NTM88 samples are programmed only with:

- Trim coefficients from FD40h to FDFFh that must not be erased. It is important to preserve the whole page from FC00h to FDFFh and ensure that the application does not erase the page;
- Protection register configured to FFh, security register configured to 82h, and reserved bytes[2] in portions of FLASH from FFDCh to FFDFh.
  When the application reprograms the interrupt vectors, the whole page from FE00h to FFFFh is erased. The erasure implies the protection and security registers are set to FFh. A security register set to FFh secures the device and prevent any further BDM access. The user must ensure that the application code configures the security register to an unsecure status value, for example to value 82h.

The rest of the flash is left empty. For NTM88 samples, only the library model can be used since no firmware functions are programmed.

In NTM88, the section from FC00h to FD3Fh is indicated as “one-time programmable”. NXP does not program this section and the section may be programmed one time since it is empty. However, after "one-time programing," the user cannot erase the section. Flash may only be erased 512 bytes by 512 bytes. Erasing the FC00h to FD3Fh section would erase the trim coefficients section which must not be done. Erasing the trim coefficients section results in non-functional sensors.

The NTM88 firmware function TPMS_FLASH_WRITE prevents write access between FD40h and FDFFh. TPMS_FLASH_WRITE may be used to write bytes between FC00h and FD3Fh with NTM88. In contrast, the FXTH firmware write access is forbidden in the whole trim page between FC00h and FDFFh. With both FXTH and NTM88, TPMS_FLASH_ERASE function cannot erase the trim page from FC00h to FDFFh.

Figure 1 shows the memory map of FXTH87/87E and NTM88.

2 Reserved bytes means that the location may not be empty when the devices are shipped by NXP, but can be erased and overwritten by the customer application.

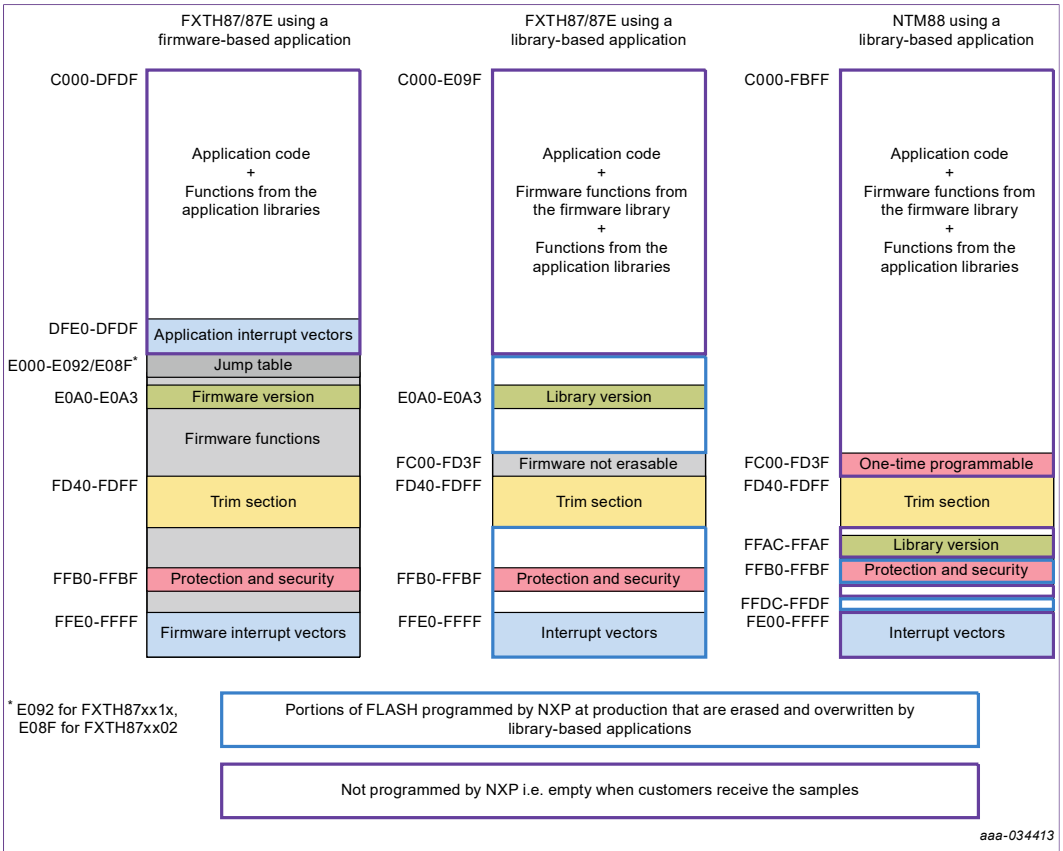

Figure 1. Memory map of the FXTH87/87E and NTM88 using firmware and library model

5 Firmware version and derivative descriptor bytes

For FXTH87 and FXTH87E, the firmware version and the derivative descriptor bytes are documented as CODE0 and CODE1 in the data sheet. The firmware version CODE0 is programmed by NXP at production with the version of the embedded firmware. When using library model applications, the library version overwrites the firmware version.

For NTM88, they are documented in the data sheet as CODEF and CODEH and are placed at different addresses. The byte CODEF is not programmed by NXP at production. It contains the firmware library version after the user has programmed an application in the device.

NXP recommends customers always use the function TPMS_READ_ID to read these bytes. NXP does not recommend a direct read from memory because the location in FXTH87/87E is different from NTM88 and NXP may change the location.

6 Firmware and application libraries

NXP provides functions to ease customer software application development. For FXTH87/87E, the functions are available in the embedded firmware (firmware model) or firmware library (library model), in the Configurable library and the new LFOCAL library.

For NTM88, the FXTH87/87E functions have been adapted for NTM88 and are available, however, some functions are placed in different libraries. Table 3 summarizes where to find the functions for all families.

Table 3. Function availability: firmware model vs. library model

	FXTH87 and FXTH87E *(firmware or library model)*	**NTM88** *(library model)*
Firmware functions listed in the FXTH87/87E firmware user guide	Available in the Embedded Firmware (firmware model) or the FXTH Firmware Library (library model)	Available in the NTM88 Firmware Library, documented in the NTM88 firmware user guide
Functions to take faster acceleration measurements (for algorithms)	Available in the FXTH Configurable Library	Available in the NTM88 Firmware Library
Functions to operate the Free Running Counter	Available in the FXTH Configurable Library	Available in the NTM88 Application Library
New LFOCAL library functions	Available in the FXTH New LFOCAL Library	Available in the Firmware Library
Function TPMS_FLASH_CHECK and functions to calculate a 16-bit CRC	Available in the Embedded Firmware (firmware model) or the FXTH Firmware Library (library model)	Source code available in separate c and header files, provided with the firmware and application libraries.

7 Reset vector and stack allocation

The reset vector used by the FXTH87 and FXTH87E devices, '_FXTH87x6LibStartup()' is defined in the FXTH firmware (embedded firmware and firmware library). Inside this function the program performs the following:

- Configuration of LF registers LFCTRLB to LFCTRLE with NXP recommended values: the user does not have to configure these registers again when initializing the LF block;
- Initialization of the stack pointer to address 28Fh (end of the RAM): the user must ensure that the stack is always allocated at the end of the RAM section;
- Jump to main.

In FXTH87/87E firmware-based applications, NXP programs the firmware reset vector at production to '_FXTH87x6LibStartup()'.

In FXTH87/87E library-based applications, the user configures the reset vector either in the *prm* file or in the interrupt vector array. The reset vector must be configured to '_FXTH87x6LibStartup()' to match the behavior of firmware-based applications. NXP library-based demo projects configure the reset vector in this manner.

For NTM88, a different solution is implemented. The project configures the default CW startup function '_Startup()' from the file *Start08.c* to be the reset vector. Inside this function the program performs the following:

- Initialization of the stack pointer to the actual address of the stack allocated by the linker: the user does not have to handle the stack allocation, it can be placed at any address by the linker.
- Jump to main.

Note: *If the preprocessor macro __ONLY_INIT_SP is not defined in the project, the _Startup function will also perform global variable initialization, which would overwrite all global variables upon exit from STOP1. This would also apply to the variables located in the RAM section preserved in STOP1. In order to avoid such a situation, make sure that the preprocessor macro __ONLY_INIT_SP is defined in the project. For that, always select the option "Minimal Startup Code" when creating a project from scratch, or manually add the macro in the project by going to Properties > C/C++ Build > Settings > HCS08 Compiler > Preprocessor.*

In the default CW startup function, the LF registers, LFCTRLB to LFCTRLE, are not configured with the recommended values. The LF registers are configured in the application code by calling the function 'vnfLFRConfigFactoryRegs()'. Refer to the NTM88 demo project for an implementation example.

Table 4. Reset vector and stack allocation summary

	FXTH87 and FXTH87E	NTM88
Reset vector	_FXTH87x6LibStartup() It is defined in the embedded firmware/firmware library	_Startup() It is the default CW startup function defined in *Start08.c*
Address at which the stack pointer is initialized in the reset vector	28Fh The user must ensure that the stack is always allocated at the end of the RAM from address 28Fh	End of stack address given by the linker. The user does not have to handle stack allocation.
LF registers LFCTRLB to LFCTRLE configured in the reset vector?	Yes	No. The LFCTRLB to LFCTRLE registers must be configured in the application code by calling the function vnfLFRConfigFactoryRegs()

8 TPMS_INTERRUPT_FLAG update

The variable TPMS_INTERRUPT_FLAG[3] holds interrupt information that occurs in several blocks. The variable must be updated in the corresponding interrupt vectors.

For FXTH87 and FXTH87E using a firmware model, the update of TPMS_INTERRUPT_FLAG is done in firmware interrupt vectors not accessible by the user application. This update is transparent to the user.

For FXTH87 and FXTH87E using a library model, there is only one set of interrupt vectors with no distinction between firmware and user interrupt vectors. The TPMS_INTERRUPT_FLAG update is done inside the interrupt handler functions defined in the firmware library. These interrupt handler functions must be called inside the appropriate interrupt vectors.

For the NTM88 using a library model, the user must update TPMS_INTERRUPT_FLAG in the interrupt vector.

3 Refer to the Firmware User Guide.

In all cases, the user must clear the register flag using the interrupt acknowledge.

Table 5. TPMS_INTERRUPT_FLAG summary

	FXTH87 and FXTH87E using firmware model	FXTH87 and FXTH87E using library model	NTM88 using library model
TPMS_INTERRUPT_FLAG update	Done inside the firmware interrupt vector, so transparent to the user.	Done inside the interrupt handler function defined in the firmware library. The interrupt handler must be called inside the interrupt vector.	To be done by the user in the interrupt vector.
Interrupt acknowledge (to clear the register interrupt flag)	To be done by the user in the user interrupt vector.	To be done by the user in the interrupt vector.	To be done by the user in the interrupt vector.

9 TPMS_FLASH functions

9.1 FXTH TPMS_FLASH functions

In FXTH firmware-based applications, TPMS_FLASH_WRITE and TPMS_FLASH_ERASE have access to user flash from C000h to DFFFh only. The function TPMS_FLASH_CHECK checks the integrity of the embedded firmware programmed by NXP at production.

In FXTH library-based applications, TPMS_FLASH_WRITE and TPMS_FLASH_ERASE have access to the whole flash except the trim page between FC00h and FDFFh.

The firmware function TPMS_FLASH_CHECK is provided to check the integrity of the trim section.

9.2 NTM88 TPMS_FLASH functions

In NTM88 library-based applications, the function TPMS_FLASH_WRITE has access to the whole flash except the trim section from FD40h to FDFFh. TPMS_FLASH_WRITE may be used to write to the one-time programmable section from FC00h to FD3Fh.

The function TPMS_FLASH_ERASE can erase any memory page except the trim page from FC00h to FDFFh. The one-time programmable section from FC00h to FD3Fh cannot be erased. Flash memory is only erased page by page, 512 bytes by 512 bytes. Erasing the FC00h to FDFFh section would erase the trim coefficients, which must not be done.

The function TPMS_FLASH_CHECK is provided to check the integrity of the trim section. It can be used with all NTM88 compatible part numbers. See "TPMS_FLASH_CHECK" in Table 6.

Table 6. TPMS_FLASH function summary

	FXTH87 and FXTH87E using firmware model	FXTH87 and FXTH87E using library model	NTM88 using library model
TPMS_FLASH_ WRITE	Access allowed between C000h and DFFFh.	Access forbidden only between FC00h and FDFFh.	Access forbidden only between FD40h and FDFFh
TPMS_FLASH_ ERASE	Access allowed between C000h and DFFFh.	Access forbidden only between FC00h and FDFFh.	Access forbidden only between FC00h and FDFFh.
TPMS_FLASH_ CHECK	Provided to check embedded firmware integrity.	Provided to check trim integrity.	Provided to check trim integrity, for compatible part numbers only.[1]

[1] Part numbers that can **not** use TPMS_FLASH_CHECK are:

- PTM88H05 with CODEH=15h or 95h
- PTM88H06 with CODEH=16h or 96h
- PTM88H13 with CODEH=1D or 9D
- PTM88H14 with CODEH=1E or 9E

All other PTM88 and NTM88 can use the function.

10 Registers and bit fields names

Compared to the FXTH87 and FXTH87E, some registers, and bit fields in NTM88 have been modified or renamed. Refer to UM11227, *NTM88 family of tire pressure monitor sensors* user manual for the complete list of the NTM88 registers.

The main modifications between the FXTH and NTM88 register definitions in CodeWarrior are listed in Table 7. Note that the list is not exhaustive and is subject to change with future CodeWarrior patch updates.

Table 7. Main modifications between FXTH and NTM88 register definitions in CodeWarrior

FXTH CodeWarrior header file	Modification	NTM88 CodeWarrior header file
KBISC_KBMOD	Bit field renamed	KBISC_KBIMOD
—	Register added	IRQSC
TPM1SC	Register renamed	TPMSC
TPM1CNTH	Register renamed	TPMCNTH
TPM1CNTL	Register renamed	TPMCNTL
TPM1MODH	Register renamed	TPMMODH
TPM1MODL	Register renamed	TPMMODL
TPM1C0SC	Register renamed	TPMC0SC
TPM1C0VH	Register renamed	TPMC0VH
TPM1C0VL	Register renamed	TPMC0VL
TPM1C1SC	Register renamed	TPMC1SC
TPM1C1VH	Register renamed	TPMC1VH
TPM1C1VL	Register renamed	TPMC1VL
—	Register added	PWUSR
—	Bit field added	PWUDIV_WDIV6 and WDIV7

FXTH CodeWarrior header file	Modification	NTM88 CodeWarrior header file
—	Bit field added	PWUSC0_WUT6 and WUT7
PWUCS0_WUFAK	Bit field moved and renamed	PWUSR_WUFACK
PWUCS0_WUF	Bit field moved	PWUSR_WUF
—	Bit field added	PWUCS1_PRST6 and PRST7
PWUCS1_PRFAK	Bit field moved and renamed	PWUSR_PRFACK
PWUCS1_PRF	Bit field moved	PWUSR_PRF
—	Bit field added	PWUS_CSTAT6 and CSTAT7
PWUS_PSEL	Bit field moved	PWUSR_PSEL
LFCTL1_LPAGE	Bit field removed	—
LFCTRLE, LFCTRLD, LFCTRLC, LFCTRLB, LFCTRLA are paged	Register unpaged	LFCTRLE, LFCTRLD, LFCTRLC, LFCTRLB, LFCTRLA are not paged
LFCTL4_DECEN	Bit field renamed	LFCTL4_DCEN
LFCTRLB_LFFAF_LFCAF0	Bit field renamed	LFCTRLB_LFCAF
LFCTRLB_LFFAF_LFCAF1	Bit field renamed	LFCTRLB_LFFAF
LFS_LFIACK	Bit field renamed	LFS_LFIAK
LFDATA_RXDATA0 to RXDATA7	Bit field renamed	LFDATA_LFRXD0 to LFRXD7
RFCR2_EOM	Bit field moved	EPR_EOM
RFCR2_RPAGE	Bit field removed	—
EPR_PLL_LPFSTR	Register renamed	EPR
RFD0 to RFD15 on page 0 and RFD0 to RFD15 on page 1	Register unpaged and renamed	RFTX0 to RFTX31
SRS	Register renamed	SIMRS
SRS_LVD	Bit field renamed	SIMRS_LVR
-	Bit field added	SIMRS_SOFT
SIMOPT1_TRH and TRE	Bit field removed	—
—	Bit field added	SIMOPT1_SPIEN
SPMSC3	Register renamed	PMCSC3
—	Bit field added	SIMSES_FRCF
SIMSES_TRF	Bit field removed	—
SIMTST	Register removed	—
SOTRM	Register renamed	SIMOTRM
SBDFR	Register renamed	SIMC
FCMD_FCMD7	Bit field renamed	FCMD_FTMR

11 Abbreviations

Table 8. Abbreviations

Acronym	Description
IDE	Integrated Development Environment
BDM	Background Debug Mode

12 Glossary

Table 9. Glossary

Term	Definition
Library Model Applications	Applications that erase the embedded firmware programmed by NXP at production. There is no distinction anymore between user flash and firmware flash. The firmware functions are added in the application via a firmware library added in the project.
Firmware Model Applications	Applications that keep the whole flash separate between user flash containing the user application and firmware flash containing the embedded firmware programmed by NXP at production. The firmware functions are called in the application code via a jump table.

13 Revision history

Table 10. Revision history

Revision	Date	Description
2	20191210	• Section 2, revised the second paragraph removing the word "production". • Section 4, revised the first and third paragraphs and Table 2. • Section 4.1, revised as follows: – Revised all paragraphs. – Revised the image and caption for Figure 1. – Removed Figure 2, captioned "Memory map of the PTM88 and NTM88 using library model." • Section 5, revised the first and second paragraphs. • Section 6, revised the last row of Table 3 and removed note before table. • Section 7, added a note after the last bullet. • Section 9.2, revised entire section including Table 6. • Section 10, revised entire section.
1	20190722	Initial release

14 Legal information

14.1 Definitions

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

14.2 Disclaimers

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

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

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

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

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

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

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

14.3 Trademarks

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

CodeWarrior — is a trademark of NXP B.V.

NXP — is a trademark of NXP B.V.

Tables

Figures

Contents

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

Date of release: 10 December 2019
Document identifier: AN12524